НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____ *Сергій, СТІРЕНКО*

«___»_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Сервіс для створення скорочених посилань»

Виконав:
студент IV курсу, групи ІО-64
Мельничук Віталій Ігорович _____

Керівник:
Ассистент кафедри ОТ
Каплунов Артем Володимирович _____

Консультант з нормоконтролю:
Професор кафедри ОТ, д.т.н.,
Сімоненко Валерій Павлович _____

Рецензент:

Асистент кафедри СПСКС,

_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій, СТІРЕНКО

«___»_____2020 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Мельничуку Віталію ігоровичу

1. Тема проєкту «Сервіс для створення скорочених посилань», керівник проєкту Каплунов Артем Володимирович, ассистент кафедри ОТ, затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту *26 травня 2020р.*

3. Вихідні дані до проєкту *див. технічне завдання*

4. Зміст пояснювальної записки *Аналіз і характеристика об'єкта проектування, обґрунтування оптимального варіанта реалізації мети цієї роботи, розробка додатку: вибір технологій та їх обґрунтування, основні рішення з реалізації додатку, тестування програми. Висновки.*

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) *приниципова схема, функціональна схема, структурна схема*

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	Сімоненко В. П., професор, д.т.н.		

7. Дата видачі завдання 01.09.2019

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Затвердження теми роботи*	*01.09.2019*	
2.	*Вивчення та аналіз завдання*	*15.12.2019-15.03.2020*	
3.	*Розробка архітектури додатку*	*15.03.2020-25.03.2020*	
4.	*Написання програмної частини*	*25.03.2020-05.04.2020*	
5.	*Тестування та виправлення помилок*	*05.04.2020-15.04.2020*	
6.	*Оформлення пояснювальної записки*	*15.04.2020-20.05.2020*	
7.	*Захист програмного продукту*	*25.04.2020*	
8.	*Передзахист*	*26.05.2020*	
9.	*Захист*	*18.06.2020*	

Студент В. І. Мельничук

Керівник А. В. Каплунов

Анотація

Дипломна робота присвячена розробці сервісу по скороченню посилань з використанням веб-фреймворків.

В роботі досліджено сучасні особливості скорочення посилань, було визначено головні вимоги до ергономічності веб-сайту.

Технологію для розробки було обрано Django для серверної частини програми, та Vue.JS для клієнтської. Зв’язок з веб-сервісом здійснено на основі архітектури REST

Створений в ході веб-сайт побудований за патерном MVT і складається з 2 модель, 3 представлень та 3 шаблонів.

Загальний обсяг роботи: 73 сторінки, 26 рисунків, 1 таблиць, 12 посилань.

Ключові слова: веб-сайт, посилання, скорочення, Django, Vue.JS

Аннотация

Дипломная работа посвящена разработке сервиса по сокращению ссылок с использованием веб-фреймворков.

В работе исследованы современные особенности сокращения ссылок, были определены главные требования к эргономичности сайта.

Технологию для разработки было выбрано Django для серверной части программы, и Vue.JS для клиентской. Связь с веб-сервисом осуществлено на основе архитектуры REST

Созданный в ходе веб-сайт построен по паттерну MVT и состоит из 2 моделей, 3 представлений и 3 шаблонов.

Общий объем работы: 73 страницы, 26 рисунков, 1 таблиц, 12 ссылок.

Ключевые слова: сайт, ссылки, сокращения, Django, Vue.JS

Abstract

This Diplom thesis is dedicated to the development of a service to reduce links using web frameworks.

In the work, the modern features of link reduction were investigated, the main requirements for the ergonomics of the site were identified.

The technology for development was chosen by Django for the server side of the program, and Vue.JS for the client. Communication with the web service is based on the REST architecture

The website created during the course is built on the MVT pattern and consists of 2 models, 3 views and 3 templates.

Total amount of work: 73 pages, 26 figures, 1 tables, 12 links.

Keywords: site, links, acronyms, Django, Vue.JS

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6415. 02.000 ТЗ	Технічне Завдання	4	
3	А4	ДП 6415. 03.000 ПЗ	Пояснювальна записка	60	
4	А4	ДП 6415. 04.000 Д1	Принципова схема	1	
5	А4	ДП 6415. 05.000 Д2	Функціональна схема	1	
6	А4	ДП 6415. 06.000 Д3	Структурна схема	1	

Зм.		№ документа	Підп.	Дата	ДП.6415.01.000 ВП		
Розробив		Мельничук В. І.			Сервіс для створення скорочених посилань	Літ.	Аркуш
Перевір.		Каплунов А. В.				Т	1 / 1
Н.конт		Симоненко В.П.				НТУУ «КПІ імені Ігора Сікорського», ФІОТ Група ІО - 64	
Затв.							

Технічне завдання
до дипломного проєкту

на тему «Сервіс для створення скорочених посилань»

ЗМІСТ

1.НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: «Сервіс для створення скорочених посилань».

Область застосування: програма може бути використана у вільному доступі будь-яким користувачем для створення скорочених посилань

2.ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського дипломного проекту, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського».

3.МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є створення сервісу для створення скорочених посилань.

4.ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література, публікації в спеціалізованих періодичних виданнях, довідники по платформах дистанційного навчання, публікації в мережі Інтернет по даній темі.

5.ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

Додаток, що розробляється повинен мати:

- Незалежність від платформи
- Можливість переглядати статистику
- Авторизація та реєстрація косритувачів
- Прикладний програмний інтерфейс

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення необхідної літератури	19.02.2020
Складання і узгодження технічного завдання	06.03.2020
Написання вступної частини та огляд рішень	19.03.2020
Розробка архітектури додатку	03.04.2020
Написання програмної частини	10.04.2020
Тестування та виправлення помилок	01.05.2020
Оформлення документації дипломного проекту	15.05.2020
Попередній захист та проходження нормативного контролю	29.05.2020
Захист дипломного проекту	15.06.2020

Пояснювальна записка
до дипломного проєкту

на тему: «Сервіс для створення скорочених посилань»

Київ – 2020 року

ЗМІСТ

ВСТУП

Сьогодні Інтернет поширений майже серед всього населення незважаючи на вік, країну та стать. Кожен день сотні мільйонів користувачів Інтернету заходять на різні веб-сайти. Інколи хочеться поділитись цікавим веб-сайтом з другом. Для цього можна просто скопіювати посилання на сайт та відправити другу в якийсь мессенджер. Але деколи посилання бувають дуже великими, понад 100 символів. Коли відправляєш 1-2 посилання, то особливих проблем немає, але коли відправляєш 5-6 посилань, тоді стає важко читати повідомлення, тому що воно забите довгими посиланнями. Особливо це стосується тих, хто ведуть блоги або відео-блоги. Зазвичай всі посилання вони лишають в кінці, або в опису, а таких посилань зазвичай багато, більше 10. Через це, опис до якогось відео стає неможливим для ознайомлення

Також коли ви публікуєте довгі посилання в соціальних мережах, це може виглядати, як спам, і користувачі просто не будуть переходити по таких посиланнях.

Для вирішення таких проблем створили спеціальні сервіси, які можуть скоротити посилання. Можна так сильно їх скоротити, що стає можливим запам’ятати посилання напам’ять, яке спочатку було більше 100 символів.

Багато засобів скорочення посилань також дозволяють відслідковувати ефективність і аналітику кожного посилання. Це допомагає зрозуміти, який контент працює найкраще на конкретних платформах.

1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття

1.1.1 URL та DNS

Уніфікований локатор ресурсів (англ. Uniform Resource Locator, URL)[2] - система уніфікованих адрес електронних ресурсів або стандартизована адреса певного ресурсу.

Доменна система імен (англ. Domain Name System, DNS)[1] – комп'ютерна система для отримання інформації про домен. Зазвичай IP-адрес по імені хоста.

Щоб краще зрозуміти як працює DNS[13] – розглянем ситуацію: Користувач вводить в адресному рядку браузера якийсь URL, наприклад YouTube.com. Що відбудеться і як браузер покаже головну сторінку сайту на який ми хочемо попасти?

Браузер складається з багатьох компонентів, один із яких є User Interface (Те, що бачить користувач). Адресний рядок це одна із частин цього компоненту. User Interface після вводу URL передає управління компоненту Browser Engine, який відповідає за взаємодію різних компонентів браузера.

Щоб зробити запит по вказаному URL потрібно знати IP сервера. Першим ділом браузер перевіряє в свій локальний кеш DNS. Компонент Browser Engine має доступ до цього кеша.

Якщо там нема відповідного запису, то браузер передає управління операційній системі, яка перевіряє свій кеш DNS. Якщо і там відсутній запис, то ОС дивиться в локальні хости. Якщо запис відсутня, то операційна система звертається до свого інтернет провайдера, у якого в наявності є свій кеш DNS, дальше йде на корінний DNS, у якого теж є свій кеш, дальше по ланцюгу серверів DNS.

Якщо на любому етапі знаходиться потрібна запис, то вона зберігається в всіх кешах і повертається браузеру. Отримання таким чином IP адреса називають DNS lookup.

Дальше Browser Engine перевіряє чи є ця сторінка в локальному кеші. Якщо немає, то передає управління компоненту Rendering Engine, який звертається до компонента Networking Component, щоб той зробив GET запит на указаний IP на порт 80 по протоколу HTTP або на порт 443 по протоколу HTTPS.

На сервері запит приймає веб сервер. В конфігураціях веб сервера прописані дозволені хости. Веб сервер перевіряє чи хост із заголовка запита host співпадає з тим, що в конфігураціях. Якщо співпадає, то сервер дивиться правила обробки такого запита і виконує їх. Наступні дії сервера залежать від технології і особливостей додатка. Припустимо що сервер згенерував HTML файл і відправив його браузеру.

Дальше браузер отримує HTML файл і компонент Rendering Engine починає розбір документа HTML.

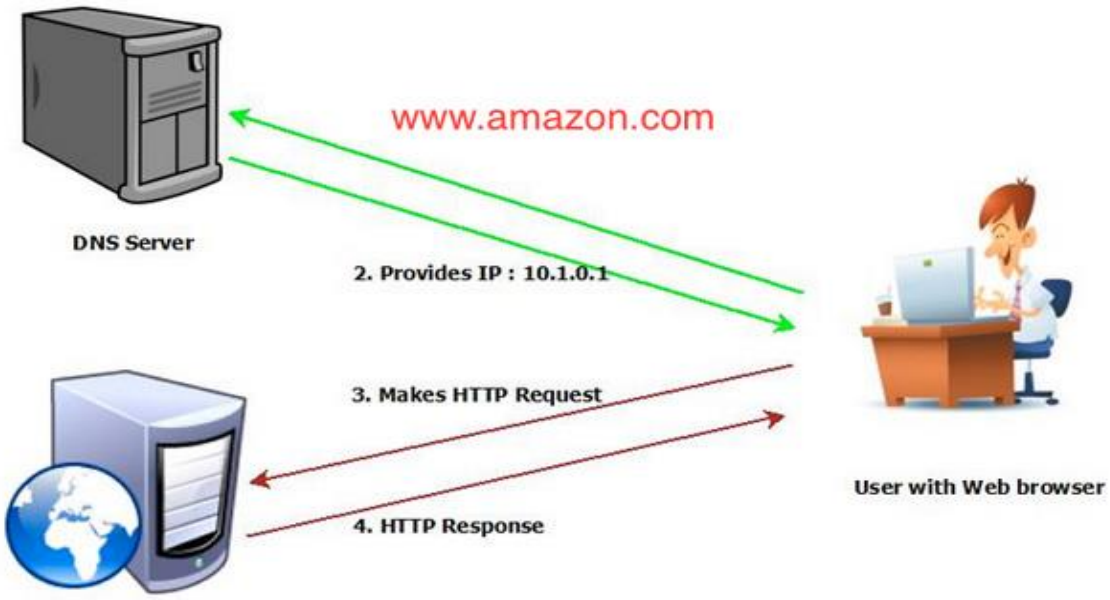


Рис. 1.1 Схема роботи DNS.

1.1.2 Як працюють скорочені URL

Скорочені посилання використовуються, щоб отримати додатковий або альтернативний URL.

Працює це все за простим принципом. Кожне “довге” посилання асоційоване з унікальним ключем, який складається з доменного імені та унікального кода. Наприклад https://domain.com/Abc123. Abc123 - унікальний ключ, який може буде згенерований, наприклад через base 36 або base62 і оснований на лічильнику, який збільшується на 1 при додаванні кожного посилання.

Коли користувач надсилає “довге” посилання на сервер. То сервер спочатку проводить перевірку, чи є посилання правильним і відповідає нормам. Якщо так, то генерує унікальний ключ.

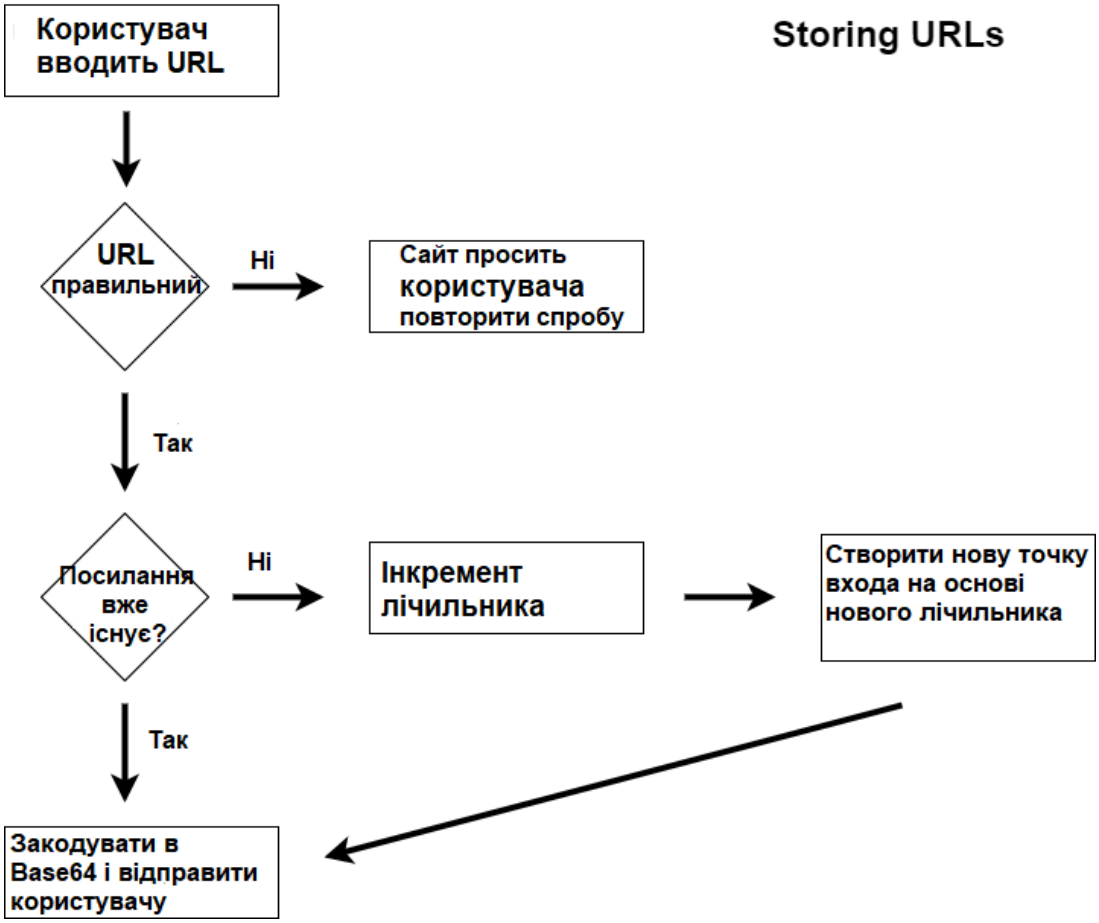


Рис. 1.2 Схема роботи скорочених посилань[4]

1.2 Основні особливості створення веб сайтів

Десять-двадцять років назад людство спокійно жило без інтернету. Але зараз життя без інтернету важко собі уявити. Кожен день сотні мільйони людей відвідують мільярди сайтів. Інтернет став незамінним майже в всіх сферах людської діяльності. На початку всіх задовольняв звичайний текстовий інтерфейс без інтерактивних елементів, але зі збільшенням кількості користувачів якість веб сайтів збільшувалась, поступово сайти стали більш інтерактивними, більш красивими, більше анімації або унікальних особливостей. І Інтернет ще досі дуже швидко розвивається.

Необхідність створення сайтів може бути різною, але в сучасному світі зараз немає компанії, в якої немає власного сайту. Багатьом приватним фізичним особам просто необхідні власні сайти для роботи, так що створення веб сайтів зараз одна із самих перспективних діяльностей.

1.2.1 Створення веб-сайтів «вручну»

Цей метод найважчий і доступний далеко не кожному. створення сайтів “вручну” базується на реалізації за допомогою HTML та CSS та якоїсь мови програмування.

Щоб знати, як розробити сайт “вручну”, потрібно розуміти, що собою представляє сайт і з чого він складається.

За відображженя сайту відповідає HTML (англ. HyperText Markup Language – мова розмітки гіпертексту) — спеціальна мова яка описує вміст сторінки. HTML підказує браузеру, що саме треба відображати коли користувач звертається до сайту. Тому, щоб створити веб сайт потрібно створити одну чи кілька HTML сторінок і поєднати їх гіперпосиланнями.

Щоб сайт виглядав красиво, потрібно ще використовувати CSS(англ. Cascading Style Sheets – каскадна таблиця стилів) — спеціальна мова, що використовується для опису зовнішнього виду веб сайту.

Для того щоб мати можливість оновлювати і керувати сайтом, сайт створюється з використанням одного із багатьох мов програмування (Python, JS, PHP, C#, Java).

Сучасний сайт має відображатись не тільки на екрані комп'ютера, але й на малих екранах телефонів чи планшетів. Цього можна досягнути, використовуючи для сайту адаптивний дизайн. Такий дизайн вміє підлашовуватися під розмір пристрою, на якому його переглядають. Створення такого дизайну потребує спеціальних навичок.

Ще одна особливість створення сайту вручну, це складність в масштабуванні та підтримці. Чим більшим буде сайт, тим важче буде працювати з ним. А під'єднати сторонні модулі буде майже неможливо.

Зазвичай спочатку створюється шаблон сторінки сайту, наприклад в Adobe Photoshop, дальше верстка сайту на HTML

1.2.2 CMS

CMS (англ. Content Management System)[11] – це система управління контентом, набір скриптів для створення, редагування і управління контентом сайту. На професійному жаргоні CMS також називають "движок". Прикладами CMS є WordPress, Joomla, OpenCart.

Якщо раніше більшість сайтів були статичними і вимагали внесення правок в їх вміст вручну то зараз динаміка розвитку проектів вимагає готовності швидко реагувати на зміни і впроваджувати їх з максимальною оперативністю. При цьому не всі користувачі хочуть або можуть собі дозволити звертатися до розробників, особливо якщо сайт вимагає постійної роботи над ним.

В свою чергу, системи управління контентом дозволяють користувачам, які не володіють навичками розробки сайтів і знаннями мов програмування, самостійно працювати над створенням і зміною сайту.

Суть роботи веб сайту полягає в схемі розділення контенту сайту та його дизайном. Користувачу дають можливість вибрати шаблон , який визначає вміст сторінки. Залишиться тільки наповнити його інформацією. Більшість

систем управління контентом грунтуються на використанні візуального редактора WYSIWYG (англ., What You See Is What You Get – що бачиш, то і получиш) — програми, які дозволяють за допомогою інтуїтивно зрозумілого інтерфейсу доставляти чи міняти інформацію на сайті. Добавлений контент зберігається в базі даних.

Як правило, CMS використовуються для таких сайтів:

- блог, форум
- інтернет-магазин
- соціальні мережі
- персональні сайти
- корпоративні сайти
- портали

Тим не менше, більшість CMS гнучко налаштовуються і можуть бути використані для різних напрямів.

Переваги CMS:

- Простота і зручність в використанні
- Доступний широкий функціонал за рахунок розширень
- Сайт можна створити за малий проміжок часу
- Наявність детальної документації

Недоліки CMS:

- Не підходять для нетипових завдань
- Популярні CMS мають одинакові вразливості
- Високе використання ресурсів при використанні плагінів
- Розширення функціоналу або вирішення важких технічних завдань потребує звернення до розробника

1.2.3 SaaS-платофрми

Програма як послуга (англ. Sofware as service, SaaS)[9] – модель поширення програм споживачам по підписці, при якій постачальник розробляє веб

сайт, розміщує і управляє ним з метою використання замовником через інтернет. Замовник платить не за розробку чи володіння таких сайтів, а за використання.

Найпростіший приклад SaaS – це Google Docs, безкоштовний сервіс для роботи з документам. Ніяких носіїв драйверів чи установок. Заходите по посиланні і працюєте з текстом, з таблицями або з презентаціями прямо в браузері. При цьому вам можуть допомагати одночасно друзі чи колеги.

По суті, SaaS це єдине програмне ядро, яке надається в використання клієнтам, доступ до системи вони отримують через мережу і можуть міняти налаштування на свій вибір. Обслуговуванням сервісу повністю займається провайдер послуги, а користувач лише працює в ній.

Популярність такої моделі щорічно росте. Розповсюджене явище, коли SaaS сервіси використовуються бухгалтерами для їх обліку та комунікації з клієнтами, редагування зображень, тобто для всього

1.2.4 REST архітектура

REST (Representational state transfer)[3] — це стиль архітектури програмного забезпечення для розподілених систем, таких як World Wide Web, який, як правило, використовується для створення веб служб. Термін REST вперше використаний в 2000 році Роєм Філдінгом, одним із авторів HTTP-протокола. Системи, які підтримають REST – називають RESTful-системами

В загальному випадку REST є дуже простим інтерфейсом управління інформацією без використання якихось додаткових внутрішніх шарів. Кожна одиниця інформації однозначно визначається глобальним ідентифікатором, таким як URL. Кожна URL в свою чергу має строго заданий формат.

Відсутність додаткових внутрішніх шарів означає передачу даних в тому ж вигляді, що і самі дані. Тобто ми не завертаємо дані в XML, як це робить SOAP і XML-RPC, ми не використовуємо AMF, як це робить Flash. Просто віддаємо самі дані.

Кожна одиниця інформації однозначно визначається URL – тобто це означає, що URL по суті є ключем для одиниці даних. Тобто наприклад третя

книжка з полиці буде мати вигляд book/3, а 35 сторінка в цій книзі — book/3/page/35. Звідси і виходить строго заданий формат. При цьому зовсім не важливо в якому форматі знаходяться дані по адресу book/3/page/35 – це може бути і HTML, і відсканована копія в вигляді jpeg-файлу, і документ Microsoft Word.

Як проходить управління інформацією сервісу — це повністю ґрунтується на протоколі передачі даних. Найбільш розповсюджений протокол це HTTP. Для HTTP дії над даними задаються за допомогою методів (GET, PUT, DELETE, POST, PUT)

Сама архітектура REST не прив'язана до конкретних технологій і протоколів, але в реаліях сучасного Веб, побудова RESTful API майже завжди має на увазі використання HTTP і будь-яких поширених форматів представлення ресурсів, наприклад JSON, або, менш популярного сьогодні, XML.

Переваги REST:

- Відсутність внутрішніх шарів, передаються самі дані
- Кожна одиниця інформація визначаються URL
- Масштабування взаємодії компонентів системи
- Незалежність компонентів
- Спільність інтерфейсів
- Проміжні компоненти, які збільшують безпеку

Щоб розподілена система вважалась сконструйованою по REST архітектурі (RESTful) необхідно, щоб вона задовільняла таким критеріям[14]

Client-Server. Система повинна бути розділена на клієнтів і на серверів. Поділ інтерфейсів означає, що, наприклад, клієнти не пов'язані зі зберіганням даних, яке залишається всередині кожного сервера, так що мобільність коду клієнта покращується. Сервери не пов'язані з інтерфейсом користувача або станом, так що сервери можуть бути простіші і легші в масштабуванні. Сервери та клієнти можуть бути замінені і розроблятися незалежно, поки інтерфейс не змінюється.

Stateless. Сервер не повинен зберігати будь-якої інформації про клієнтів. У запиті повинна зберігатися вся необхідна інформація для обробки запиту і якщо необхідно, ідентифікація клієнта.

Cache. Кожна відповідь повинна бути зазначена чи є вона кешувальною чи ні, для запобігання повторного використання клієнтами застарілих або некоректних даних у відповідь на подальші запити.

Uniform Interface. Єдиний інтерфейс визначає інтерфейс між клієнтами і серверами. Це спрощує і відокремлює архітектуру, яка дозволяє кожній частині розвиватися самостійно.

Як видно, в архітектура REST дуже проста в плані використання. По виду запиту, який прийшов відразу можна визначити, що він робить, не розбираючись в форматах (на відміну від SOAP, XML-RPC). Дані передаються без застосування додаткових шарів, тому REST вважається менш ресурсовитратним, оскільки не треба “парсити” запит щоб зрозуміти що він повинен зробити і не треба переводити дані з одного формату в інший

1.3 Аналіз Back-End фреймворків

Стандарти веб-розробки постійно зростають разом зі складністю сучасних технологій. Винахід власних “велосипедів” тепер займає надто багато часу і сил. На допомогу розгубленому розробнику поспішають фреймворки.

Але тут з'являється нова проблема: цих фреймворків дуже багато. В кожного є свої переваги та недоліки. Зараз порівняємо їх[15]

1.3.1 Express

Почнемо опис з самого простого фреймворку, який використовується на платформі Node.js.

Express використовується для розробки додатків досить давно і завдяки своїй стабільності міцно займає позицію одного з найпопулярніших фреймворків Node.js.

Для цього фреймворка існує велика кількість докладних інструкції та описів, які складені розробниками, перевірити його ефективність на практиці.

Погодьтеся, набагато розумніше скористатися накопиченим і перевіреним досвідом, ніж заново винаходити “велосипед”.

Основна особливість цього фреймворка полягає в тому, що для Express характерний невеликий обсяг базового функціоналу. Всі інші необхідні вам функції потрібно буде добирати за рахунок зовнішніх модулів. По суті, Express в чистому вигляді - це сервер і у нього може не бути жодного модуля.

Завдяки такому мінімалізму розробник спочатку отримує в своє розпорядження легкий і швидкий інструмент, який він може розширювати і розвивати.

При цьому важливо, що вибір модулів для Express не пов'язаний ні з якими обмеженнями: ні з кількісними, ні з функціональними.

В результаті, цей фреймворк забезпечує розробнику можливість вирішувати будь-які завдання, не обмежуючи його при цьому у виборі засобів.

З одного боку, не може не радувати той факт, що відсутність готових універсальних рішень фактично означає, що кожне створюване додаток буде унікальним.

З іншого боку, розробнику потрібно самостійно відбирати і організовувати модулі, а це передбачає великий обсяг роботи і відповідно, вимагає від розробника більше часу і зусиль.

Переваги Express:

- Простота
- Гнучність
- Легко масштабувати
- Детальна документація
- Великий набір додаткових модулів
- Велика спільнота

Недоліки Express:

- Великий об’єм ручної роботи
- Старий підхід на основі callbacks функцій

1.3.2 Rail

Популярний Ruby-фреймворк з класичною структурою Model-View-Controller. Rails успішно працює в Airbnb, GitHub, Hulu і Shopify.

Інструмент лояльний до новачків і має невисокий початковий поріг входження. Однак за сценою там чимало магії, варто зробити кілька перших кроків, і доведеться дертися на круту гору.

Щоб зробити роботу з фреймворком швидше і ефективніше, створено безліч корисних “ґемів” (gems, пакети і бібліотеки), які можна підключити до вашого додатком. Rails-спільнота досить сильне і доброзичливе, крім того в мережі є чимало навчальних ресурсів по цьому інструменту.

Переваги Rails:

- Швидкість розробки на 40-50% більша ніж на інших мовах програмування
- Висока гнучкість
- Наслідування. Завдяки цьому взаємодія між розробниками краща
- Ефективність затрат

Недоліки Rails:

- Важкий в навчанні
- Проблеми з хостінгом. Надто дорого
- Досить мало документації

1.3.3 Laravel

Laravel - це безкоштовний PHP фреймворк з відкритим вихідним кодом, створений Тейлором Отвеллом для розробки веб-додатків за архітектурним шаблоном MVC.

Можна сказати, що на появу Laravel вплинули інші PHP фреймворки.

Він був створений як альтернатива фреймворку Codeigniter, в якому було недостатньо корисних функцій для розробки веб-додатків. В якості основи Laravel виступають компоненти іншого фреймворка - Symfony

Фреймворк Laravel дуже популярний серед західних розробників веб-додатків.

За допомогою менеджера пакетів Composer, фреймворк Laravel дозволяє легко встановлювати і підключати різні компоненти для використання в веб-додатку.

Реалізація шаблону ActiveRecord - Eloquent ORM, дозволяє встановити відносини між об'єктами бази даних веб-додатки і вибудовувати зручні запити для маніпуляції даними.

Механізм автозавантаження класів дозволяє не підключати вручну файли через include але й запобігає завантаженню що не використаних компонентів.

У Laravel є вбудована підтримка движка шаблонів Blade, за допомогою якого можна робити прості уявлення веб-додатки використовуючи спеціальний синтаксис.

У Laravel є багато корисних функцій, що дозволяють зробити процес розробки веб-додатків швидким, простим і якісним.

Плюси Laravel:

- Досить зрозуміла документація
- Велика екосистема
- Гнучка система маршрутизації
- Багато різних функцій
- Надійна система захисту баз даних

Недоліки Laravel:

- Немає зворотної сумісності між різними версіями
- Нелогічне розташування каталогів та файлів
- Мало перекладу документації

1.3.4 Django

Коли у вас виникає певна ідея, трансформувати її на мові програмування і надати їй реальну форму за допомогою Django займе всього кілька хвилин.

Те, що Django знаходиться у вільному доступі, дає можливість помітно зменшити процес веб розробки, так як розробник може сфокусуватися на процесі дизайну і розробки функціоналу програми.

Таким чином, Django - це ідеальний інструмент для стартапів, коли веб дизайн повинен відображати концепцію і цілі компанії.

Django з'явився в 2005 році, і поступово став одним з кращих фреймворків, який допомагав і допомагає тисячам розробників виконувати ту чи іншу роботу протягом декількох хвилин. Спочатку Django був фреймворком для мови Python, з відмінним функціоналом, Django помітно спростив ряд складнощів в розробці веб додатків, надавши цій роботі більш спрощений підхід

Переваги Django:

- Простота в вивченні
- Чистота коду
- Швидкість написання
- Безпека
- Різносторонність
- Масштабованість

Недоліки Django:

- Використання шаблону маршрутизації із зазначенням URL
- Надто монолітний
- Все базується на ORM Django
- Компоненти розгортаються спільно
- Необхідно вміння володіти всією системою для роботи

1.4 Аналіз Front-End фреймворків

1.4.1 Angular

Angular створений для спрощення складних процесів створення та управління JS додатками. В основі лежить структура MVC, що робить фреймворк особливо корисним для створення односторінкових сайтів. Бібліотека заснована на звичайному JS і HTML, тому Angular автоматично піклується про маніпуляції з DOM і AJAX запитах, які в іншому випадку розробникам довелося б писати самим. Інструмент надає модульні будівельні блоки коду JS, які можна поєднувати і тестувати. Angular можна швидко додати на будь-яку HTML сторінку за допомогою простого тега.

Плюси Angular:

- Велика кількість різноманітних функцій
- Функції взаємозалежні
- Можливість працювати окремо в одному розділі програми
- Мінімальний ризик помилок

Недоліки Angular:

- В Основі важка мова програмування(TypeScript)
- Помилки під час міграції між версіями

1.4.2 Vue

Розпочавшись як проект одного розробника Google, Vue.js дуже швидко виріс в один з найпопулярніших JavaScript-фреймворків.[12]

Це дуже гнучкий інструмент з прогресивною структурою, який легко інтегрувати в уже існуючі проекти. Компонентна архітектура і багата екосистема дозволяє розробляти складні додатки з мінімальними витратами.

Плюси Vue:

- Велика та функціональна бібліотека
- Можна використовувати на любому проекті
- Займає мало місця

- Висока швидкість розробки
- Низькій поріг входу

Недоліки Vue:

- Не такий зручний підхід компонентів як в React
- Шаблонізація React більш гнучкіша
- Не найкраща система рендерінга

1.4.3 React

Не дуже правильно називати React фреймворком, це скоріше бібліотека компонентів для веб-розробки. Однак його значення таке велике, що історично жодне порівняння без нього не обходиться.

Саме React від Facebook ввів "моду" на компонентну архітектуру і віртуальний DOM.

Розробка ведеться на особливому діалекті JavaScript - JSX. Це суміш звичного JS з таким же звичним HTML. І в цілому це дуже інтерфейс-орієнтований інструмент, істотно спрощує роботу з веб-сторінкою в браузері.

React можна використовувати не тільки на клієнті, але і на стороні сервера.

Плюси React:

- В основі проста мова програмування
- Велика гнучкість додатку
- Використовує DOM
- Додаток витримує великі навантаження
- Має відкриту бібліотеку даних
- Невеликий обсяг бази даних
- Проста міграція між версіями

Недоліки React:

- Незручна документація
- Надто великий вибір інструментів
- Потрібно багато часу, щоб вивчити всі нюанси

1.5 Огляд альтернатив

1.5.1 Bit.ly

Bit.ly[16] - платформа для скорочення посилань з великою панеллю інструментів, на якій відображаються показники ефективності ваших посилань: рейтинг “кліків”, статистика каналу і географічні дані користувачів. Крім того, є можливість інтеграції з програмним забезпеченням для управління соціальними мережами, такими як Sprinklr, Sprout Social, Buffer, Hootsuite і HubSpot, щоб допомогти розміщувати скорочені посилання через ваші профілі в соціальних мережах.

Безкоштовний обліковий запис Bit.ly пропонує до 500 фірмових посилань, 10000 не брендових посилань і дані звітів, що ідеально підходить для малих підприємств. Корпоративний план платформи дозволяє клієнтам маркувати необмежену кількість посилань, і надає всі дані і показники, згадані вище. Корпоративний тариф найкраще підходить для великих підприємств, які хочуть маркувати і відстежувати кожне посилання в своїх маркетингових кампаніях.

1.5.2 Bl.ink

Bl.ink є одним з найбільш надійних сервісів скорочення посилань. З його допомогою можна створювати розумні фірмові посилання, що містять релевантні слова, а не просто випадкову рядок символів. Сервіс також надає аналітичні звіти, які можуть відстежувати кліки по даті, часу, мові, посиланнях, влаштуванню і розташуванню і інтегруватися з інструментами веб-аналітики, такими як Google Analytics, Adobe і іншими. Bl.ink пропонує чотири рівні підписки: безкоштовний план, план для окремих осіб і невеликих груп з кількістю користувачів до 15, план для груп і підприємств з числом ко-

ристувачів до 50 і план для великих організацій з числом користувачів більше 50. Наприклад, безкоштовні користувачі можуть створювати до 1000 посилань і відстежувати до 1000 кліків по посиланню. Після цього ви будете платити в залежності від того, скільки посилань ви створюєте і відстежуєте.

1.5.3 Rebrand.ly

Rebrand.ly, сервіс, з яким працюють понад 250 000 клієнтів, є платформою для скорочення посилань, яка може маркувати посилання, відстежувати їх показники ефективності або інтегруватися з більш ніж 50 іншими платформами для безперешкодного поширення ваших посилань. З двома планами для приватних осіб і двома планами для команд, Rebrand.ly пропонує рішення по скороченню посилань для компаній малого і великого бізнесу. Наприклад, їх початковий план пропонує 50 000 кліків, 5000 фірмових посилань і 5 призначених для користувача доменних імен за 29 доларів в місяць, а їх преміальний план пропонує 2 000 000 кліків, 200 000 фірмових посилань і 20 призначених для користувача доменних імен за 449 доларів на місяць.

1.5.4 Ow.ly

Ow.ly, сервіс, розроблений платформою управління соціальними мережами Hootsuite, включений в кожен безкоштовний аккаунт Hootsuite. За допомогою Ow.ly ви можете розміщувати посилання і відслідковувати показники їх ефективності безпосередньо на платформі Hootsuite. Це дозволяє скоротити кожну опубліковану вами посилання на всі ваші профілі в соціальних мережах. Ow.ly найкраще підходить тим, хто вже використовує Hootsuite в якості платформи управління соціальними мережами.

1.5.5 Buff.ly

Подібно інструменту скорочення посилань Hootsuite, Buff.ly інтегрований в Buffer: ще одну платформу управління соціальними мережами. За допомогою Buff.ly ви можете скорочувати свої посилання, налаштовувати їх, поширювати по всіх своїх профілях в соціальних мережах і відстежувати їх показники ефективності прямо на платформі Buffer. Buff.ly ідеально підхо-

дить для тих, хто використовує Buffer в якості платформи управління соціальними мережами.

1.6 Актуальність та різниця між альтернативами

В Компактна, візуально приваблива скорочене посилання - невід'ємна частина рекламної кампанії для будь-якого проекту в інтернеті. Короткий адреса посилання добре запам'ятовується, крім того їм зручно ділиться з іншими користувачами. Використовуючи стислий URL можна відстежувати статистику, в тому числі аналізувати потік відвідувачів.

З 30 березня 2018 року Google оголосив закриття сервісу googl. Даний сервіс працював з 2009 і користувався популярністю серед звичайних користувачів інтернету і веб-розробників. Це був спеціальний сервіс для скорочення посилань. З 13 квітня 2018 роки тільки існуючі користувачі могли скористатися наявними можливостями googl, але до 30 березня 2019 року сервіс остаточно перестав існувати. Створені посилання зберегли своє перенаправлення.

Чому закрили цей сервіс? Для подальшого розвитку нового сервісу під назвою Firebase Dynamic Links (FDL). FDL - це інтелектуальні URL-адреси, які дозволяють відправляти існуючих і потенційних користувачів в будь-яке місце в iOS, Android або веб-додаток.

Але все ж перед відданими користувачами, які активно використовували функціонал сервісу googl постав питання, чим його замінити і які аналоги більш зручні. Отже, зараз дуже актуальні різні сервіси по скороченню посилань

Різниця мого проекту між конкурентами в тому, що, по-перше, мій сервіс безкоштовний, а всі решта платні, якщо використовувати, наприклад, більше 1000 кліків.

Також, на відміну від конкурентів, де скорочене посилання генерується автоматично, в моєму сервісі буде можливість самому вибрати назву посилання. Ще в мене буде реалізовано API для розробників. Тобто до мого серві-

су можна буде звертатись не тільки через веб сайт, але й через dekstop-додаток або мобільний додаток

Висновки до першого розділу

У цьому розділі я провів функціональний аналіз вимог до серверної частини та клієнтської частини додатку. Було наведено список різних технологій, бібліотек та фреймворків, які максимально спрощували та покращували реалізацію функціоналу, який потрібен для мого проекту.

Також було проаналізовані загальні принципи побудови веб сайтів. Різні підходи та архітектури та структури сайтів.

Скорочені посилання потрібні не тільки заради зручності їх використання, легкості в зберіганні, але й для збору різних видів статистики, наприклад, порівняння ефективності реклами, або визначати цільову аудиторію.

2. ВИБІР ЗАСОБУ РЕАЛІЗАЦІЇ

2.1 Функціональні вимоги до системи

Одною з важливіших частин розробки кожної системи це проектування архітектури, тому що архітектура, яка розроблена правильно, гнучко, значно покращує майбутню підтримку, а наступні проблеми будуть виявлені значно швидше.

Тобто, при виборі інструментів та технологій потрібно бути дуже уважним та обережним. Потрібно вибирати не тільки ті технології, які зручні саме вам, але й ті, які потрібні проекту. Теж потрібно враховувати швидкість розробки, чистоту коду та швидкість вивчення технології.

Технології мають вибиратись за наступними ознаками:

- Система взаємодіє з Базою Даних, значить потрібна підтримка взаємодії з БД.
- Система має REST архітектуру. Тобто це Restful система. Отже вибрані технології повинні максимально спрощувати обмін запити будь-якими клієнтами та сервером
- Веб сайт буде в вигляді SPA. Тому потрібен фреймворк, який дозволяє легко розробляти сайти такого типу

Також потрібною частиною проектування це визначення структурних частин архітектури. Частини системи мають бути незалежні одна від одної.

Щоб побудувати таку архітектуру потрібно чітко визначити функціональні вимоги системи.

Розроблена серверна частина системи повинна підходити під наступні вимоги:

- легка взаємодія з JavaScript клієнтами через систему запитів
- Створення API Endpoint та системи маршрутизації
- Можливість переглядати дані з БД та вносити зміни в неї
- Переадресація клієнта

Розроблена клієнтська частина системи повинна підходити під наступні вимоги:

- Простий та зручний графічний інтерфейс користувача
- Можливість реєстрація та аутентифікації
- Перегляд статистики
- Взаємодія з серверною частиною через JSON

2.2 Мова програмування

Система по скороченню посилань повинна мати Rest архітектуру, Значить при виборі мови програмування потрібно звертати увагу на те, чи придатна мова для того, щоб робити клієнт-серверні додатки.

Я вибрав для цієї цілі мову програмування Python.

На відміну від C#, ця мова програмування більш популярна, отже легше знайти подібну проблему в інтернеті.

На відміну від Java, код на Python виглядає значно чистіше. 50 рядків на Java, це 10 на Python.

Головна різниця від PHP, це те, що в Python на даний момент одна стабільна версія 3.x, а в PHP зараз найкраща версія це 7.2, але немає документація детальної та підручників чи книг по цій версії, приходиться вчитись по старим версіям, які не є досконалими.

А JavaScript досить хороша технологія, але важко розгорнути її без зайвих проблем. А якщо потрібно писати якісний код, то тоді прийдеться вчити TypeScript, який дуже важкий для вивчення.

Отже мова програмування Python ідеально підходить для цього проекту, і її будемо використовувати.

2.3 Використані бібліотеки та фреймворки

Фреймворки полегшують життя розробнику, пропонуючи різні рішення для розробки додатків і сервісів. Вони автоматизують впровадження стандартних рішень, а відтак скоротити час. Таким чином, розробник фокусується на додатку, а не на рутинних завданнях, де не потрібно творче мислення.

Python - один з найпопулярніших і простих для вивчення мов програмування і застосовується майже всюди, в тому числі в веб-розробці. Для нього є багато фреймворків, частина яких не просто полегшує розробку, але і надає інструменти, що дозволяють буквально за пару днів підняти готовий сайт.

Такі фреймворки відносяться до категорії full stack. Вони потужні, в них багато інструментів і все включено, але це може зробити їх важкими, повільними і негнучкими

Головний плюс фулстек-фреймворків в тому, що все, потрібне для повноцінного додатка, в них вже є. Не потрібно шукати окремі бібліотеки для кожного дрібного завдання і думати про сумісність, тому навіть початківці зможуть швидко зібрати готовий додаток.

2.3.1 Django

Серед бібліотек або фреймворків для серверної частини додатку я вибрав фреймворк Django

Фреймворк Django написаний на мові програмування Python, тому його структура відповідає особливостям мови. Розробники реалізували в Django шаблон MVC, і він застосовується в поточній версії фреймворка.

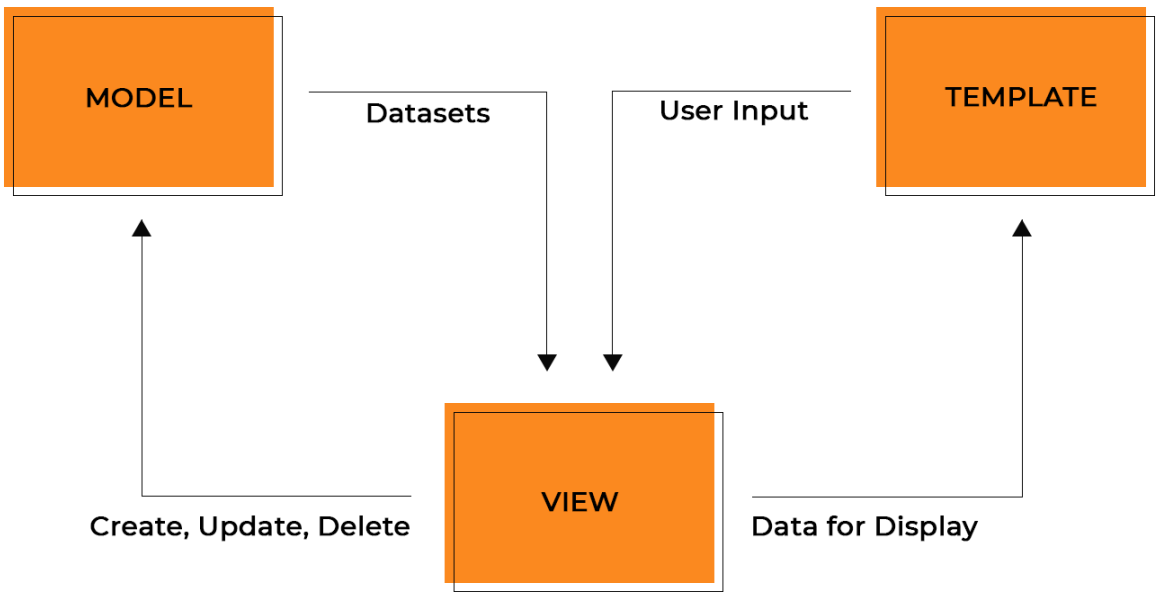


Рис 2.1 - Схема структури MVT в Django[6]

Архітектура MVC дозволяє розробнику працювати з візуальним представленням і бізнес-логікою додатка окремо. До речі, при роботі з Django фахівці частіше використовують термін MVT - Model-View-Template або модель-уявлення-шаблон. Компоненти MVT можна використовувати незалежно один від одного.

Документація Django визначає модель (model) як «джерело інформації про дані, в яких містяться ключові поля і поведінку даних». Зазвичай одна модель вказує на одну таблицю в базі даних. Django підтримує бази даних PostgreSQL, MySQL, SQLite і Oracle.

Моделі містять інформацію про дані. Ці дані представлені атрибутами або полями. Оскільки модель являє собою простий клас, вона нічого не знає про інших рівнях Django. Взаємодія між рівнями відбувається через API.
Модель відповідає за бізнес-логіку, методи, властивості і інші елементи, пов'язані з маніпуляцією даними. Також моделі дозволяють розробникам створювати, читати, оновлювати та видаляти об'єкти в базі даних.

Подання (view) вирішує три завдання: приймає HTTP-запити, реалізує бізнес-логіку, визначену методами і властивостями, відправляє HTTP-відповідь у відповідь на запити. Тобто відображення отримує дані від моделі і надає шаблонами (templates) доступ до цих даних або попередньо обробляє дані і потім надає до них доступ шаблонами.

В Django реалізований потужний движок шаблонів і власну мову розмітки. Шаблони являють собою файли з HTML-кодом, за допомогою якого відображаються дані. Вміст файлів може бути статичним або динамічним. Шаблони не містять бізнес-логіки. Тому вони тільки відображають дані.

Досвідчені розробники рекомендують сприймати Django як систему. Це означає, що фреймворк зазвичай використовується з великою кількістю сторонніх додатків. Їх можна вибирати в залежності від потреб конкретного проекту.

Django був представлений в 2005 році. За 15 років існування він сильно змінився і удосконалився. У фреймворку постійно з'являються нові можливості, а старі удосконалюються.

Адміністративна панель Django автоматично генерується при створенні програми. Це позбавляє розробника від необхідності створювати “адмінку” вручну.

Функціональність Django розширюється за допомогою плагінів. Це програмні модулі, які дозволяють швидко додати на сайт потрібну функцію. В офіційному каталозі є сотні плагінів, які дозволяють легко реалізувати на сайті, управляти доступами, підключити платіжну систему і так далі. При необхідності ви можете відключати або замінювати плагіни, щоб пристосувати її до поточних потреб проекту.

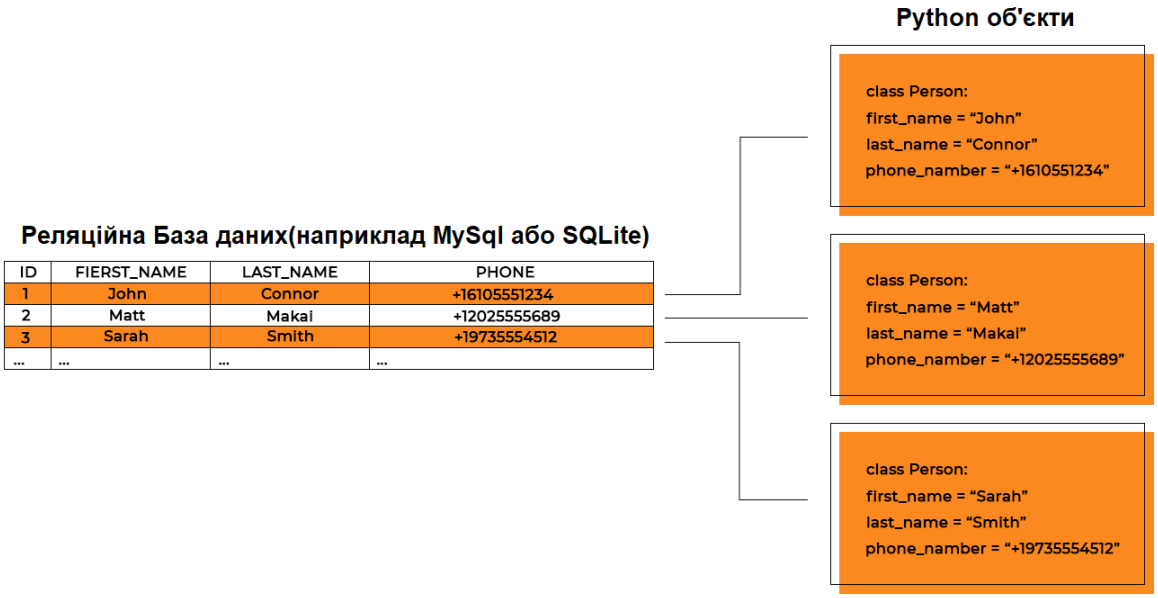


Рис 2.2 - Схема роботи ORM

В Django реалізовано об'єктно-реляційне відображення (ORM), яке забезпечує взаємодію додатка з базами даних (БД). ORM автоматично передає дані з БД, наприклад, PostgreSQL або MySQL, в об'єкти, які використовуються в коді програми.

Django підтримує використання бібліотек при розробці веб-додатків. У число популярних бібліотек входять:

- Django REST Framework, який спрощує роботу з API.
- Django CMS - зручний інструмент для управління контентом.
- Django-allauth - з його допомогою реалізуються функції реєстрації, авторизації, управління обліковими записами.

2.3.2 Vue.JS

Серед бібліотек або фреймворків для клієнтської частини додатку був вибір між Angular, React та Vue.JS. Для свого проекту я вибрав Vue.JS[8]

Vue.js - це молодий фреймворк, який був створений в 2014 році, друга версія вийшла в 2016 році.

Деякі розробники називають його бібліотекою за малий розмір і простоту використання.

Напевно це плюс, тому що розробники врахували плюси і мінуси React і Angular і створили такий собі гібрид, який увібрав в себе все найкраще з цих фреймворків і не повторював їх помилки.

Переваги Vue над React та Angular:

- Vue легше в використанні, а значить швидкість розробки більша
- в Vue одна із найкращих і повних документацій
- Vue займає менше місця. Тобто сайт завантажується швидше
- Архітектура додатку значно гнучкіша
- Vue більш продуктивний ніж Angular
- Vue найбільш популярніший фронтенд фреймворк
- Не обов’зяково використовувати vue/cl для використання
- Є можливість значно зменшити розмір бібліотеки

Таблиця 1.1.

Порівняння розміру фреймворків[12]

Фреймворк	**Розмір**
Angular 2	111K
Angular 2 + Rx	143K
Angular 1.4.5	51K
React 0.14.5 + React DOM	40K
React 15.3.0 + React DOM	43K
Vue 2.4.2	20.9K

2.3 Вибір технології взаємодії з БД

Для взаємодії з Базами Даних я буду використовувати Django ORM

ORM — технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи “віртуальну об'єктну базу даних”

Оскільки Django ORM перетворює любі об’єкти в таблиці в любій СУБД, то вибір СУБД не так важливий, але потрібний, тому що різниця все ж таки існує.

Я вибирав між SQLite та PostgreSQL. Та вибрав PostgreSQL[17]

PostgreSQL не просто реляційна, а об'єктно-реляційна СУБД. Це дає йому деякі переваги над іншими SQL базами даних з відкритим вихідним кодом, такими як MySQL, MariaDB і Firebird.

Фундаментальна характеристика об'єктно-реляційної бази даних - це підтримка об'єктів і їх поведінки, включаючи типи даних, функції, операції, домени і індекси. Це робить PostgreSQL неймовірно гнучким і надійним. Се-

ред іншого, він вміє створювати, зберігати та видавати складні структури даних. У деяких випадках ви побачите вкладені і складові конструкції, які не підтримуються стандартними РСУБД.

У PostgreSQL безліч можливостей. Створений з використанням об'єктно-реляційної моделі, він підтримує складні структури і широкий спектр вбудованих і обумовлених користувачем типів даних. Він забезпечує розширену ємність даних і заслужив довіру дбайливим ставленням до цілісності даних. Можливо, вам не знадобляться всі ті функції зберігання даних, які присутні, але, оскільки потреби можуть швидко зрости, є безсумнівна перевага в тому, щоб мати все це під рукою.

Висновки до другого розділу

На основі тих даних, які я оглянув та аналізів, які я провів в першому розділі, було вирішено, що я буду проектувати додаток з допомогою REST архітектури. Отже, для серверної частини, я вирішив, що буду використовувати мову програмування Python, та потужний веб-фреймворк Django.

А для клієнтської частини я буду використовувати мову програмування JavaScript на платформі Node.JS, та фронтед-фреймворк Vue.JS. Ці дві технології дуже просто поєднати між собою, тому що вони дуже гнучкі та потужні.

Для Баз Даних я буду використовувати СУБД PostgreSQL, тому що вона має найбільший функціонал, хоча всі таблиці і контент буде заповнюватись через Django ORM.

3.ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ

В даному розділі буде розглянуто створення сервісу по скороченню посилань за допомогою веб-фреймворку Django та JS-фреймворку Vue.JS.

Буде детально описано установку фреймворків на локальний сервер, опис та розробка Баз Даних, розробка сайту з архітектурою MVT та способи взаємодії клієнтської та серверної частини

3.1 Огляд проекту

3.1.1 Огляд бібліотек та інструментів

Для оптимальної роботи сервісу було вирішено використати наступні програми, технології та інструменти:

- Python 3.8.3 (мова програмування)
- pip 20.1 (Менеджер пакетів для Python)
- Django 3.0.5 (веб-фреймворк для серверної частини)
- Django Rest Framework 3.11.0 (Потужний інструмент для створення restful систем)
- django-rest-auth 0.9.5 (Бібліотека, яка позволяє створити реєстрацію користувача через API Endpoint)
- django-allauth 0.42.0 (Інтегрований набір програм, що стосуються автентифікації та реєстрації)
- django-registration 3.1 (Додаток, який дозволяє зробити нестандартну реєстрацію користувача)
- django-crispy-forms 1.9.1 (Потужний інструмент, для створення красивих форм)
- django-webpack-loader 0.7.0 (Інструмент, який генерує статичні пачки, без звичайних статичних файлів Django)
- npm 6.10.3 (Менеджер пакетів Node.JS)
- Vue.JS (прогресивний JS-фреймворк для створення реактивних додатків)

- @vue/cli 4.3.1 (npm пакет, який дозволяє швидко та зручно створювати Vue додатки)
- vue-router 2.6.11 (офіційний роутер для Vue.JS)
- JetBrains PyCharm Professional 2020.1.1 (Інтегроване середовище розробки
- PostgreSQL 11.4 (Система управління базами даних)

3.1.2 Огляд архітектури проекту

Сервіс побудований на MVT(Model-View-Template) архітектурі. Детальніше про неї буде в розділі 3.5. В цьому підрозділі буде огляд архітектури проекту, та опис кожного файлу та директорії

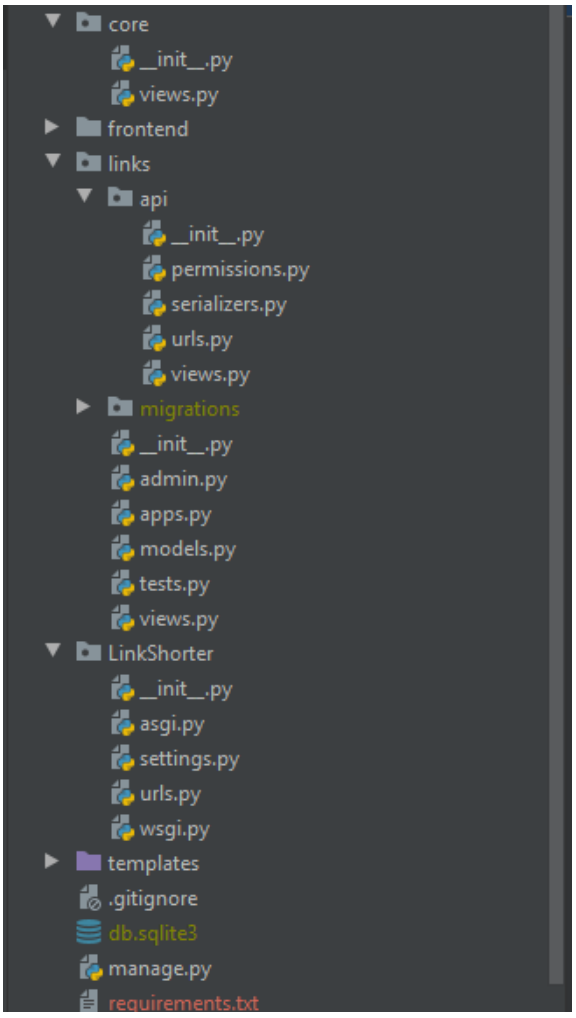


Рис 3.1 - Архітектура додатку

- settings.py – Містить в собі всі настройки проекту. Тут ми реєструємо додатки, налаштування БД і так далі.

- urls.py – Задає асоціації URL адрес з представленням. Є кілька різних urls.py, тому що прийнято ділити його на частини, по одній на додаток
- wsgi.py - Використовується для налагодження зв'язку між Django додатком і веб-сервером
- asgi.py - Асинхронний інтерфейс шлюзового сервера
- manage.py - використовується для створення додатків, робота з базами даних і для запуску сервера.
- __init__.py – Пустий файл, створений для того, щоб Python та Django розпізнавав папку як Python модуль
- migrations/ - Папка, в який зберігаються всі «міграції», це файли які дозволяють обновляти базу даних
- links/ - Додаток, який відповідає за основну логіку програми
- admin.py – Налаштування адмін-панелі в Django
- apps.py – Локальні Налаштування додатку links
- models.py – Представлення таблиць баз даних в вигляді моделі
- tests.py – Тести для додатку
- links/views.py – Представлення для додатку links
- links/api/ - Вся логіка API для цього сервісу
- seriazlizers.py – Перетворює складні типи даних, такі як набори запитів, JSON в початкові типи даних Python
- links/urls.py – Точки входу для API
- links/api/views.py – Логіка точків входу API
- permissions.py – Створення власних прав для користувачів
- templates/ - Тут розміщуються Django шаблони
- static/ - Тут будуть розміщуватись статичні файли (зображення)
- core/views.py – Точка входу Django в Vue.JS
- .gitignore – Список файлів та директорій, які має ігнорувати Git
- db.sqlite3 – Тимчасова БД. Під час розробки її використовувати зручніше, ніж PostregSQL

- requirements.txt – Список залежностей

3.2 Реалізація взаємодії з БД

Архітектура Django дозволяє значно прискорити процес розробки завдяки простої схеми використання баз даних в додатках. Django ORM надає простий механізм роботи з базою без вивчення синтаксису SQL запитів. Однак подібне абстрагування може привести до неефективного використання БД, що може позначитися на повільній роботі сайтів навіть при невеликих обсягах даних.

Постійне з’єднання покращує продуктивність, дозволяючи не створювати нове підключення до бази даних при кожному запиті. Налаштування CONN_MAX_AGE вказує як довго існує з’єднання. Це налаштування можна вказати для кожної бази даних окремо.

Django виконує підключення до бази даних при першому запиті. Він тримає з'єднання і використовує його для подальших запитів. Django закриває з'єднання по закінченню CONN_MAX_AGE, або коли воно не може бути більше використано.

Точніше, Django автоматично створюєте з'єднання з базою даних, якщо воно необхідне, і немає відкритого з'єднання, тому що це перше з'єднання, або попереднє було закрито.

На початку кожного запиту Django закриває з'єднання, якщо минув його термін. Якщо база даних закриває з'єднання після певного часу, то слід вказати менше значення в CONN_MAX_AGE, щоб Django не намагався використовувати закрите з'єднання.

В кінці кожного запиту Django закриває з'єднання, якщо минув його термін, або, якщо з'єднання знаходиться в стані невиправної помилки. Якщо в процесі обробки запиту сталася помилка бази даних, Django перевіряє чи працює з'єднання, і закриває його, якщо воно не працює. Таким чином помилка бази даних впливає тільки на один запит, для подальших запитів буде створено нове з'єднання.

Щоб почати роботу з базами Даних в Django, потрібно спочатку в налаштуваннях вказати з якою БД буде працювати Django.

Налаштування БД робиться в файлі settings.py. Для цього потрібно визначити змінну DATABASES. За замовчуванням в Django такі налаштування Баз данних:

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.sqlite3',
      'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
   }
}
```

Це означає, що Django буде використовувати СУБД - SQLite, та збережить файл БД в db.sqlite3.

SQLite - це хороша База Даних, тому що з нею легко працювати. Не треба довго її налаштовувати, Django все зробить за вас.

Але ця База Даних не підходить для серйозних проектів, тому я буду використовувати PostgreSQL. Одна з найпопулярніших та потужніших Баз Даних .

Щоб почати роботу, потрібно спочатку встановити Postgres. Його можна скачати з офіційного сайту (https://www.postgresql.org/download/). Вибираємо версію Операційної системи, в моєму випадку Windows та завантажуємо файл інсталяції.

Після того, як файл скачався, потрібно запустити його. При виборі шляху для встановлення, краще вибрати запропонований:
(C:\Program Files\PostgreSQL\10)

Зразу при установці вибираємо пароль для супер користувача (postgres) та порт за замовчуванням (5432).

Щоб перевірити, чи установка виконана правильно, потрібно ввести в командний рядок psql –V. Якщо не буде помилки, то все правильно. Мій вивід команди виглядає так:

psql (PostgreSQL) 11.4

Після успішної установки запускаємо SQL Shell (psql) та створюємо Базу Даних. Це зробити можна однією командою:

>CREATE DATABASE link_db;

Створено базу даних link_db. Тепер потрібно підключити її в наш Django додаток. Це можна зробити в налаштуваннях (settings.py). Щоб підключити, потрібно написати такий код:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'link_db',
        'USER': 'postgres',
        'HOST': '127.0.0.1',
        'PORT': 5432
    }
}
```

ENGINE – Яку СУБД ми використовуємо

NAME – Назва БД

USER – Назва користувача

Тепер Django підключений до бази даних link_db. Тепер потрібно заповнити її контентом. Базу Даних автоматично заповнює Django ORM на основі моделей, які ми напишемо в файлі models.py

Я створив дві моделі. Модель Link та Click для опису таблиць Link та Click. Вони виглядають так:

```
class Link(models.Model):
    user = models.ForeignKey(User, blank=True, null=True, default=None, on_delete=models.SET_NULL)
    long_link = models.CharField(max_length=200)
    short_link = models.CharField(max_length=200, unique=True)

    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def click_count(self):
        return len(Click.objects.filter(link=self))

    def __str__(self):
        return f"{self.click_count()} click: {self.short_link} ({self.long_link})"

class Click(models.Model):
    link = models.ForeignKey(Link, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return f"click on {self.link.short_link} at {self.created_at.strftime('%b %d %Y %H:%M:%S')}"
```

Рис 3.2 - models.py

Два Класа означають дві таблиці. Кожне поле в класі характеризує одне поле в таблиці. Наприклад поле класа Link:

long_link = models.CharField (max_length=200).

Це Створить в таблиці Link поле long_link яке буде мати тип Char з обмеженням на довжину символів 200.

Метод click_count в класі Link рахує та повертає кількість переходів по посиланню.

Щоб застосувати зміни в Базу Даних, потрібно ввести 2 команди міграції:

> python manage.py makemigrations

> python manage.py migrate

Перша команда створює міграції до всіх додатків, що вказані в налаштуванні в змінній INSTALLED_APPS, та зберігає їх в папці migrations/ . Їх можна перевірити, та змінити так, як потрібно.

Друга команда застосовує зміни, і звертається напряму до Бази Даних.

3.3 Реалізація взаємодії з клієнтською частиною

Клієнт і сервер взаємодіють один з одним в мережі Інтернет або в будь-який інший комп'ютерної мережі за допомогою різних мережевих протоколів, наприклад, IP протокол, HTTP протокол,.

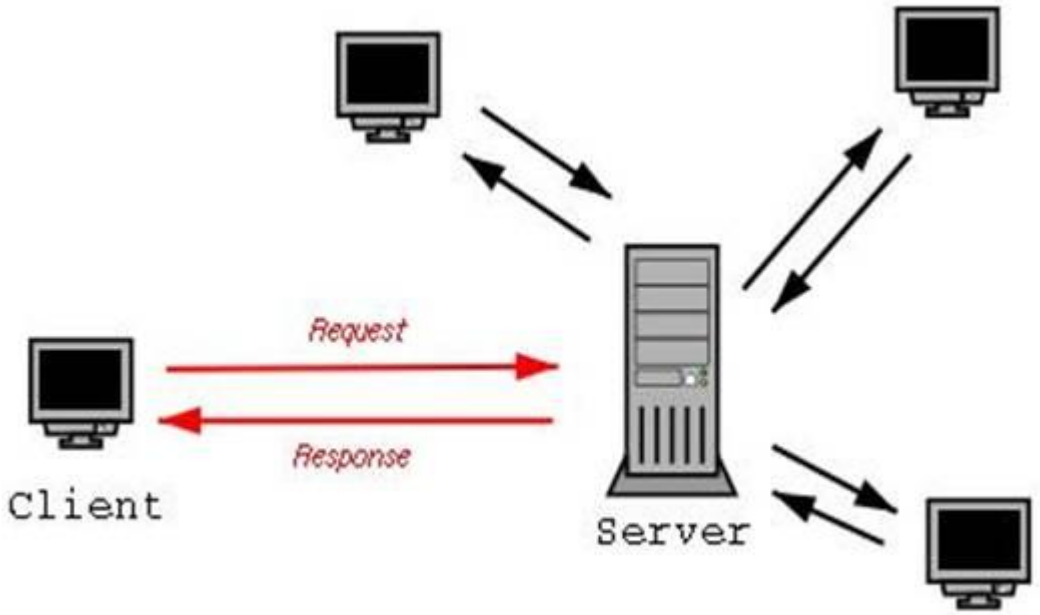


Рис 3.3 - Взаємодія Клієнтів та Сервера

Протоколів насправді дуже багато і кожен протокол дозволяє надавати ту чи іншу послугу. Наприклад, за допомогою HTTP протоколу браузер відправляє спеціальне HTTP повідомлення, в якому зазначено яку інформацію і в якому вигляді він хоче отримати від сервера, сервер, отримавши таке повідомлення, відсилає браузеру у відповідь схоже по структурі повідомлення (або кілька повідомлень), в якому міститься потрібна інформація, як правило це HTML документ.

3.3.1 Реалізація представлення об’єктів в вигляді JSON

Сервер може підтримувати кілька різних форматів для передачі одних і тих самих даних (JSON, XML) і зберігати дані в своєму внутрішньому форматі. Але найпопулярніший та найзручніший формат даних це JSON. Отже, потрібно всі дані в вигляді Python об’єктів перетворити в JSON формат[7]

Для досягнення цієї цілі я буду використовувати Серіалізатори Django

Серіалізатори дозволяють перетворювати складні дані, такі як набори запитів та екземпляри моделей, у початкові типи даних Python, які потім можуть бути легко перетворені у JSON, XML чи інші типи вмісту. Серіалізатори також забезпечують десеріалізацію, дозволяючи проаналізовані дані перетворюватися назад у складні типи після попередньої перевірки вхідних даних.

Серіалізатори в рамках REST працюють дуже аналогічно класам Django Form та ModelForm. Ми надаємо клас серіалізаторів, який дає потужний, загальний спосіб контролювати вихід ваших відповідей, а також клас ModelSerializer, який надає корисну комбінацію для створення серіалізаторів, які стосуються екземплярів моделі та запитів.

Серіалізатори створюють в файлі serializer.py. Я створив два серіалізатора для моделі Link та Click.

Кожне поле потомка класа ModelSerializer відповідає полю в моделі. Метод get_object, який генерується SerializerMethodField() зроблений для того, щоб вручну задати поведінку серіалізації відповідного поля.

Клас Meta служить для того, щоб вказати серіалізатору, які поля потрібно серіалізувати, і яку модель для цього використовувати

```
5   class LinkSerializer(serializers.ModelSerializer):
6       user = serializers.StringRelatedField(read_only=True)
7       click_count = serializers.SerializerMethodField()
8
9       created_at = serializers.SerializerMethodField()
10
11      class Meta:
12          model = Link
13          exclude = ["updated_at"]
14
15      def get_created_at(self, instance):
16          return instance.created_at.strftime("%B %d, %Y")
17
18      def get_click_count(self, instance):
19          return instance.click_count()
20
```

Рис 3.4 - Серіалізатор LinkSerializer

```python
class ClickSerializer(serializers.ModelSerializer):
    link = serializers.StringRelatedField(read_only=True)
    created_at = serializers.SerializerMethodField()

    class Meta:
        model = Click
        exclude = ["updated_at"]

    def get_created_at(self, instance):
        return instance.created_at.strftime("%B %d, %Y")
```

Рис 3.5 - Серіалізатор ClickSerializer

3.3.2 API Endpoint

Інтерфейс прикладної програми (API) дозволяє взаємодіяти двум системам. І майже з кожною установою, яка приймає стратегію API, важливо, розуміти різні аспекти та основи API та як ними керувати, щоб забезпечити найвищий рівень роботи користувачів.

Простими термінами, Endpoint(точка входу) API - це точка входу в канал зв'язку, коли дві системи взаємодіють. Це стосується сенсорних точок зв'язку між API та сервером. Точки входу можна розглядати як засіб, з якого API може отримати доступ до ресурсів, необхідних серверу для виконання їх завдання. Endpoint API - це в синонім URL-адреси сервера або послуги.

Усі ми знаємо, що API працюють через "запити" та "відповіді". І коли API запитує на доступ до даних із веб-програми чи сервера, відповідь завжди надсилається назад. Місце, куди API надсилає запит, і де відповідь надходить, називається точкою входу. Як вважається, Точки входу є найважливішою частиною документації API, оскільки саме розробник реалізує свої запити.

Оскільки все більше людей починають цінувати використання API для сприяння передачі критичних даних, транзакцій та процесів, стало важливо розуміти різні аспекти, що створюють API. Таким чином, переконання в надійності точок зв’язку між системами є важливим для успіху API. Точки входу допомагають зобразити точне розташування ресурсів, до яких має доступ

API, а також відіграють важливу роль у забезпеченні правильного функціонування програмного забезпечення, яке взаємодіє з API. Тому ефективність та продуктивність API залежить від його здатності ефективно взаємодіяти та спілкуватися з API Endpoint.

В Django Точки входу в API реалізовуються в файлі urls.py

```
router = DefaultRouter()
router.register(r"links", lv.LinkViewSet)
urlpatterns = [
    path("", include(router.urls)),
    path("links/<slug:short_link>/click/",
        lv.ClickCreateAPIView.as_view(),
        name="click-create"),
    path("links/<slug:short_link>/clicks/",
        lv.ClickListAPIView.as_view(),
        name="click-list"),]
```

За точки входу в Django відповідає змінна urlpatterns, яка включає в себе кільки шляхів (path). Один шлях, це одна точка входу, наприклад path("links/<slug:short_link>/click/", lv.ClickCreateAPIView.as_view(), name="click-create"),

Ця інструкція означає, що створено точку входу з назвою “click-create”, яка буде доступна по посиланню “links/<slug:short_link>/click/” та буде використовувати як основу представлення ClickCreateAPIView.

Конструкція <slug:short_link> означає, що тут може бути люба послідовність символів, яка задовольняє формат slug (цифри, латинські букви та тире) з можливістю доступу до неї через ім’я short_link

Також в Django Rest Framework є потужна функція router:

router.register(r"links", lv.LinkViewSet)

Ця інструкція дозволяє автоматично створити зразу кілька точків входу на основі представлення LinkViewSet. В моєму випадку вона створює 5 точків входу CRUD + L (Create Retrieve Update Delete + List). Дозволяє створювати, получати, оновлювати, видаляти дані та получати список всіх даних.

Список всіх точків входу API для мого сайту:

- api-auth/login/ - Вхід (Авторизація) через API браузера
- api-auth/logout/ - Вихід через API браузера
- api/rest-auth/login/ - Вхід(Авторизація) через REST
- api/rest-auth/logout/ - Вихід
- api/rest-auth/user/ - Детальна інформація про користувача
- api/rest-auth/password/change/ - Зміна пароля
- api/rest-auth/password/reset/ - Скидання пароля(Забули пароль)
- api/rest-auth/password/reset/confirm/ - Підтвердження скидання пароля
- api/rest-auth/registration/ - Реєстрація
- api/rest-auth/registration/verify-email/ - Підтвердження електронної почти
- api/rest-auth/registration/account-confirm-email/ Підтвердження аканта через електронну почту
- GET api/links/ - Получити список всіх посилань
- POST api/links/ - Створити нове посилання
- GET api/links/<short-link>/ - Детальна інформація про посилання
- PUT api/links/<short-link>/ - Змінити інформацію про посилання
- DELETE api/links/<short-link>/ - Видалити посилання
- POST api/links/<short-link>/click/ – Перейти по посиланню
- GET api/links/<short-link>/clicks/ - Список всіх переходів по посиланню

точки входу створюються в urls.py, але їх логіка створюється в links/api/views.py

Логіка для CRUD + L операцій для links виглядає так:

```python
class LinkViewSet(viewsets.ModelViewSet):
    """Provide CRUD +L functionality for Link."""
    queryset = Link.objects.all().order_by("-created_at")
    lookup_field = "short_link"
    serializer_class = LinkSerializer
    permission_classes = [IsAuthenticated, IsAuthorOrReadOnly]

    def perform_create(self, serializer):
        serializer.save(user=self.request.user)

```

Рис 3.6 - CRUD + L операції для link

Django Rest Framework дуже потужний інструмент. 5 точків входу створено завдяки одному класу. Цей клас унаслідується від MovelViewSet, це потужний клас, який дозволяє легко реалізувати логіку для точків входу.

LinkViewSet використовує як основу модель Link та серіалізатор LinkSerializer, а також класи прав користувача IsAuthorOrReadOnly та IsAuthenticated. Метод perform_create визначає як будуть зберігатися дані.

```python

class ClickListAPIView(generics.ListAPIView):
    """Provide the clicks queryset of a specific list instance."""
    serializer_class = ClickSerializer
    permission_classes = [IsAuthenticated, IsAuthor]

    def get_queryset(self):
        kwarg_short_link = self.kwargs.get("short_link")
        link = get_object_or_404(Link, short_link=kwarg_short_link)

        self.check_object_permissions(request=self.request, obj=link)

        return Click.objects.filter(link__short_link=kwarg_short_link).order_by("-created_at")


```

Рис 3.7 - Операція list для click

На цей раз не можна було використовувати MovelViewSet, тому що операцію видалення, операцію зміни не потрібно добавляти, тому прийдеться створити операцію list через не менш потужний клас ListAPIView, який бере за основу модель Click, серіалізатор ClickSerializer, та права користувача

IsAuthenticated та IsAuthor. Метод get_queryset визначає як получати інформацію.

```
10  class ClickCreateAPIView(generics.CreateAPIView):
11      queryset = Click.objects.all()
12      serializer_class = ClickSerializer
13
14      def perform_create(self, serializer):
15          kwarg_short_link = self.kwargs.get("short_link")
16          link = get_object_or_404(Link, short_link=kwarg_short_link)
17
18          serializer.save(link=link)
19
```

Рис 3.8 - Операція create для click

Операцію create зроблено через клас CreateAPIView. ClickCreateAPIView працює на основі моделі click, серіалізаторі ClickSerializer. Метод perform_create визначає як будуть зберігатись об’єкти

3.4 Опис форм реєстрації та авторизації

Веб-форми - це сторінки, які користувачі запитують за допомогою браузера. Ці сторінки можна написати за допомогою поєднання HTML, клієнтського скрипта, серверних елементів управління і серверного коду. Коли користувачі запитують сторінку, вона компілюється і виконується на сервері платформою, а потім платформа створює розмітку HTML, яку браузер може візуалізувати.

Для реалізації форм я використав 3 різних варіанта. Спочатку розглянемо форму через бібліотеку django-rest-auth. За допомогою неї, я реалізував форму через REST API. Тобто зареєструватись та авторизуватись в мій сервіс можна буде не тільки через веб-додаток, але через API (в теорії це може бути будь-що, мобільний додаток або комп’ютерна програма). Цей спосіб реалізовується досить просто. Потрібно просто вказати в файлі urls.py шляхи до форми реєстрації:

path("api/rest-auth/", include("rest_auth.urls"))

path("api/rest-auth/registration/", include("rest_auth.registration.urls"))

В результаті, якщо перейти по посиланню api/rest-auth/registration/ то можна побачити форму

Оскільки я доступився до неї через браузер, вона прийняла графічний інтерфейс, але якщо через командний рядок доступатись, то прийдеться через JSON формат вводити дані. Звичайно, існує така ж сама форма, але не для реєстрації, а для авторизації.

Рис 3.9 - Форма реєстрації django-allauth

Ця форма корисна лише при розробці. Користувачі зазвичай будуть користуватись наступною формою, яка реалізована зразу за допомогою двох бібліотек django-registration та crispy-forms

Наступний варіант авторизації та реєстрації розроблений завдяки бібліотеці django-allauth, яка корисна, якщо реєструватись з API браузера.

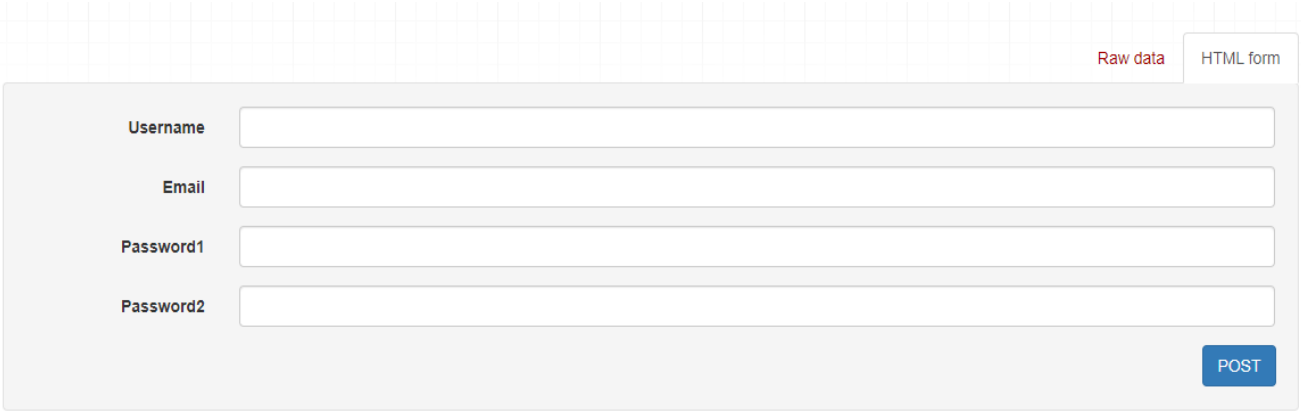


Рис 3.10 - Форма реєстрації rest-auth

django-registration відповідає за логіку та контент форм, а crispy-forms відповідає за зовнішній дизайн форми, її структуру та вигляд. Я вибрав шаблон bootstrap 4, для цього потрібно в налаштування Django вказати:

```
CRISPY_TEMPLATE_PACK = "bootstrap4"
```

Рис 3.11 - Форма реєстрації crispy-forms

Цією формою будуть користуватись звичайні користувачі. Django буде пересилати всіх неавторизованих користувачів на цю форму реєстрації.

3.5 Архітектура Vue додатку

Поділ коду стосовно односторінкового додатком є хорошим способом прискорити завантаження програми.

Так як в цьому випадку користувачеві не потрібно чекати, доки завантажиться все додаток цілком. Він може почати користуватися додатком з самого начала, як тільки почалося завантаження цього додатка.

Такий підхід покращує взаємодію користувача з додатком, особливо це стосується мобільних додатків. Цей фактор також важливий для пошукової оптимізації, так як Google знижує в пошуковій видачі сайти з низькою швидкістю завантаження.

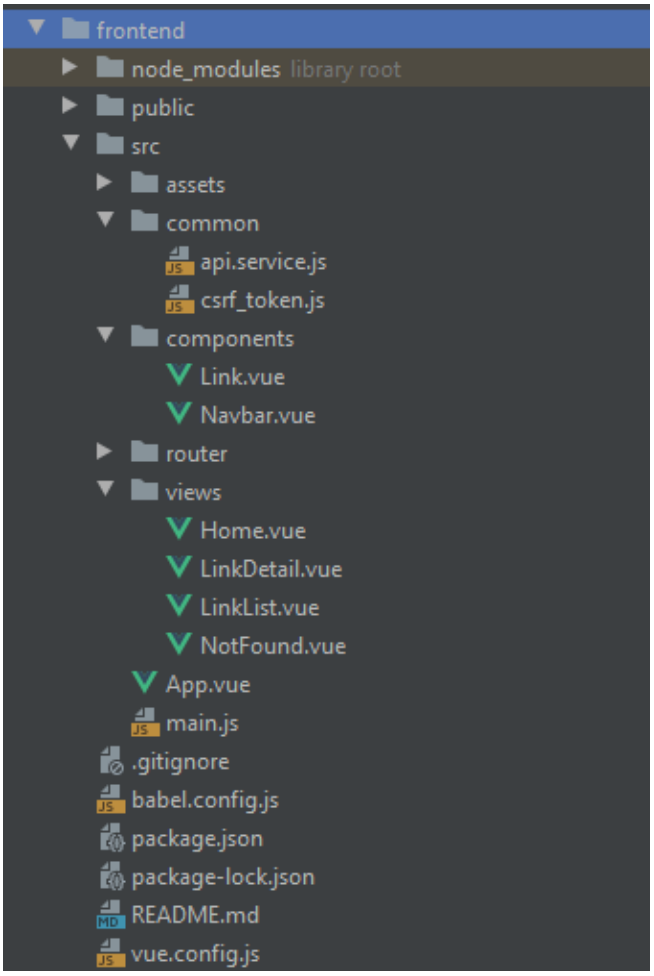


Рис 3.12 - Архітектура Vue додатка

- node_modules – Каталог, в який установлюються всі локальні пакети NPM
- package.json – Зберігає список необхідних пакетів для проекту
- public/ - Публічні статичні файли
- assets/ - Зберігає статичні ресурси та зображення
- router/ - Розширення Vue-router. Відповідає за маршрутизацію модулів та функцій.
- vue-config-js – Файл конфігурації зв’язаних з управлінням станів
- components/ - Папка включає в себе загальні компоненти, які розділені на кілька модулів функцій або компонентів.
- common/ - js функції для взаємодії з сервером
- api.service.js – Виклик HTTPS запитів для зв’язку з Django додатком
- csrf_token.js – Отримання CSRF токена, для безпечних запитів форм

- Link.vue – Компонент, який відображає 1 об’єкт посилання
- Navbar.vue – Компонент, який відображає навігаційну панель
- Home.vue – Стартова сторінка з полем вводу
- LinkDetail.vue – Детальна інформація про посилання
- LinkList.vue – Список всіх посилань та статистика їх використання
- NotFound.vue – Компонент, який буде видно, при помилці 404
- App.vue – Головний компонент, який запускається, через Django
- main.js – Налаштування головного компоненту
- .gitignore – Спеціальний файл для системи контролю версій, який вказує які файли не потрібно добавляти в git
- babel.config.js – Налаштування для транс компілятора javascript
- README.MD – Опис проекту в мові маркування
- webpack-stats.json – Файл конфігурації webpack , який потрібен для того, щоб можливо було писати Vue додаток, в різних файлах
- package-lock.json – описує залежності верхнього рівня для інших пакетів з допомогою semver

Висновки до третього розділу

Django заохочує вільне зв'язування і суворий поділ частин програми. Якщо слідувати цій філософії, то легко вносити зміни в одну конкретну частину додатки без шкоди для інших частин. У функціях відображення, наприклад важливо відділення бізнес-логіки від логіки відображення за допомогою шаблонної системи. Використовуючи шар для роботи з базою даних, ми застосовуємо цю ж філософію для логіки доступу до даних.

Ці три речі разом - логіка доступу до даних, бізнес-логіка і логіка відображення - складають концепцію, яку називають шаблоном Модель-Представлення-Шаблон (Model-ViewTemplate, MVT) архітектури програмного забезпечення. У цій концепції термін «Модель" визначає логіку доступу до даних; термін «Подання» відноситься до тієї частини системи, яка визначає, що показати і як; а термін «Шаблон» відноситься до тієї частини системи, яка визначає яке представлення треба використовувати, в залежності від призначеного для користувача введення, по необхідності отримуючи доступ до моделі.

4. ТЕСТУВАННЯ ПРОГРАМИ

Тестування програм – це процес, при якому виявляють можливі помилки, або недоробки в програмі. Незважаючи на те, що тестування не допомагає усунути всі неполадки, але воно значно зменшує кількість майбутніх помилок в коді.

4.1 Функціональне тестування програми

Для перевірки правильності роботи програми пройдемось по всім функціям.

Спочатку попробуємо зареєструватись. Для цього перш за все потрібно запустити додаток. Оскільки це локальна версія, фронтенд і бекенд запускаються окремо через команди:

>python manage.py runserver - команда запускає Django додаток

>npm run serve - команда запускає Vue додаток

Переходимо на localhost, тобто http://127.0.0.1:8000/

Одною з важливіших частин розробки кожної системи це проектування архітектури, тому що архітектура, яка розроблена правильно, гнучко, значно покращує майбутню підтримку

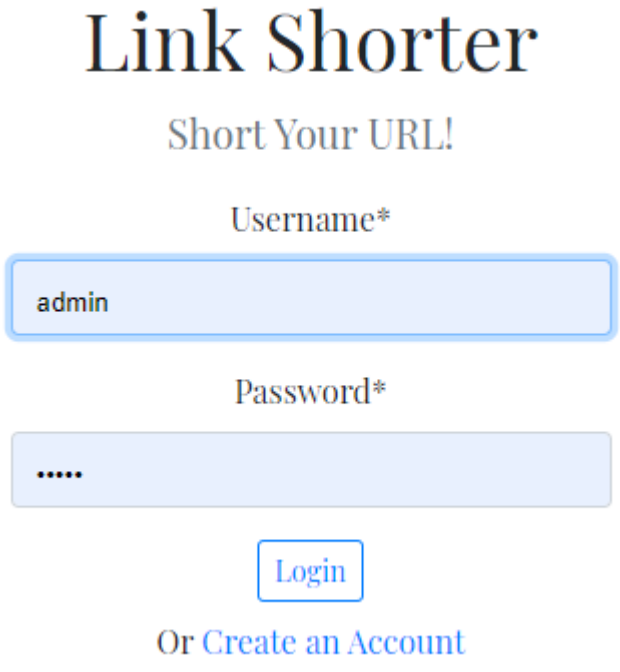


Рис 4.1 - Вікно авторизації

Для реєстрації потрібно нажати посилання Create an Account після цього з’являться 4 поля.

- Username – Ім’я користувача
- Email address – Електронна пошта користувача
- Password – Пароль, який повинен бути більше 8 символів
- Password confirmation – Підтвердження пароля

Для тесту створимо користувача testUser. Вводимо дані, нажимаємо кнопку create account. Якщо реєстрація пройде успішно, відбудеться перенаправлення на головну сторінку сайту.

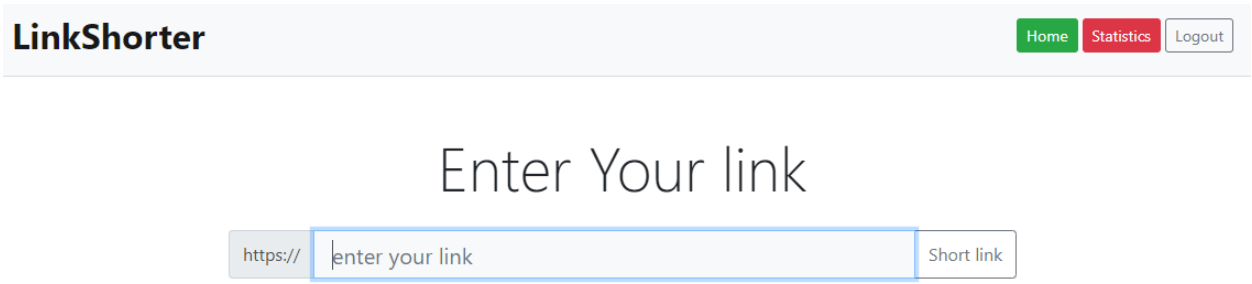


Рис 4.2 - Головна сторінка сайту

Щоб переконатись, що все працює, введемо якесь довге посилання в головну форму сайту, наприклад статтю про генератори в Python (http://masnun.com/2015/11/13/python-generators-coroutines-native-coroutines-and-async-await.html)

Після вводу посилання, нажимаємо на кнопку Short link

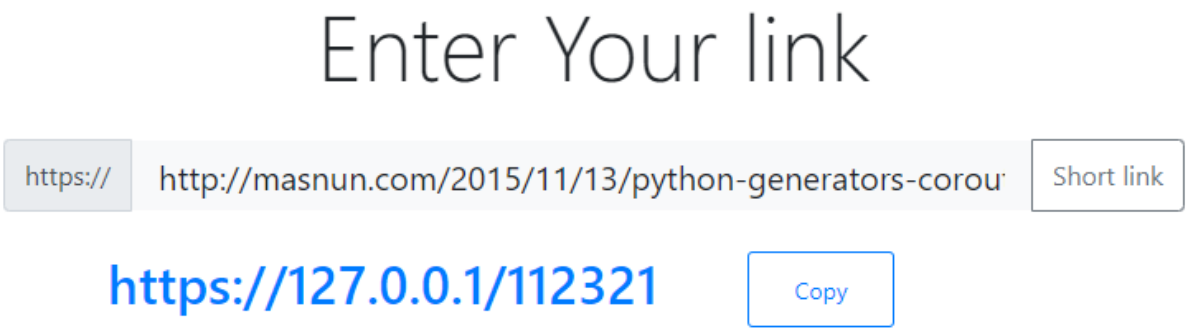


Рис 4.3 - Генерація скороченого посилання

Згенерувалось коротке посилання. Тепер можна скопіювати його, або нажати на кнопку Copy, тоді воно автоматично скопійовано в буфер обміну.

Доменне ім’я тут localhost, тому що я запускаю все локально. Якщо запустити з сервера, то буде доменне ім’я хоста

Тепер завжди, коли ми будемо переходити по даному короткому посиланню, то нас будуть пересилати на довге посилання.

Вводимо в адресний рядок браузера посилання і переходимо по ньому

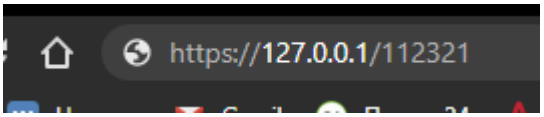


Рис 4.4 - Перехід по скороченому посиланню

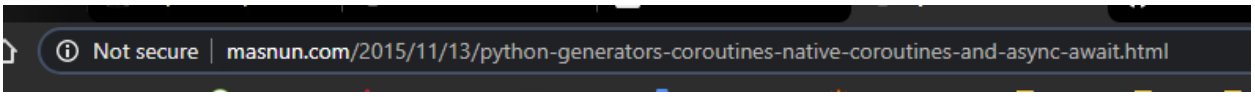


Рис 4.5 - Результат переходу по посиланню

Перехід спрацював. В Посилання повинно було збільшитись лічильник click_count. Щоб перевірити це, можна зайти в статистику, на навігаційному панелі є кнопка Statistics. Нажимаємо на неї

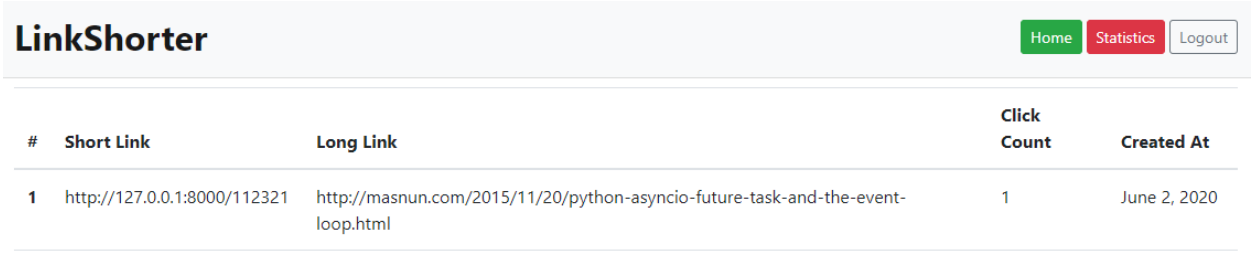


Рис 4.6 - Статистика посилань

Як Видно з рисунка, лічильник переходів спрацював, а саме посилання добавилось в базу даних, як і очікувалось.

Якщо нажати на рядок таблиці з посиланням, то можна перейти на сторінку, з детальною інформацією про посилання.

В детальну інформацію входить, довге та коротке посилання, дата створення посилання та дата кожного окремого переходу по посиланню.

Отже, щоб інформація була нагляд ніша, потрібно ще пару раз перейти по посиланню.

Після того, як я два рази перейшов по посиланню, в статистику добавилось ще 2 кліка.

link_id	short_link	long link	Click count	Created At
5	https://127.0.0.1/112321	http://masnun.com/2015/11/20/python-asyncio-future-task-and-the-event-loop.html	3	June 02, 2020

Click id	Clicked_at
8	June 02, 2020 (21:04:02)
7	June 02, 2020 (21:03:52)
6	June 02, 2020 (18:10:50)

Рис 4.7 - Детальна інформація про посилання

Як видно з детальної статистики, в кожного переходу є своя дата. Таким чином можна перевіряти в яку пору доби посилання найбільш активніше

Отже, по результатам тестування, весь функціонал працює як заплановано та без лишніх збоїв.

Висновки до четвертого розділу

В Четвертому розділі було наведено результати Тестування програми,а саме функціональне тестування. Тести були зроблені на сценаріях, які найчастіше будуть використовуватись користувачами.

Всі розроблені компоненти програми, а саме реєстрація та авторизація, запис та читання даних в Базу Даних, перегляд списку посилань та детальна статистика та інформація про посилання, відповідають нормам.

ВИСНОВКИ

Сервіс по скороченню посилань - простий і зручний інструмент, яким можна скористатися онлайн.

В представленій роботі, було розглянуто різні способи та підходи до архітектури та структури веб-додатків. Також розглянуто різні фронтенд та бекенд фреймворки. На основі аналізів, вибрано найкращі альтернативи, та було реалізовано все на через мову програмування Python та JavaScript та фреймворками Django та Vue.JS

Є 3 основні причини скорочувати посилання:

- Деякі посилання, можуть займати більше 200-300 символів, коли скорочені можуть 20-30. Якщо треба поділитись з кимось 4-5 посиланнями, то різниця дуже суттєва
- Якщо скинути не скорочене посилання, то інші користувачі можуть прийняти це як спам
- Сервіси по скороченню посилань дозволяють збирати статистику, що дозволить дізнатись, з якого джерела переходять на ваш сайт частіше, або вік та країна цільової аудиторії

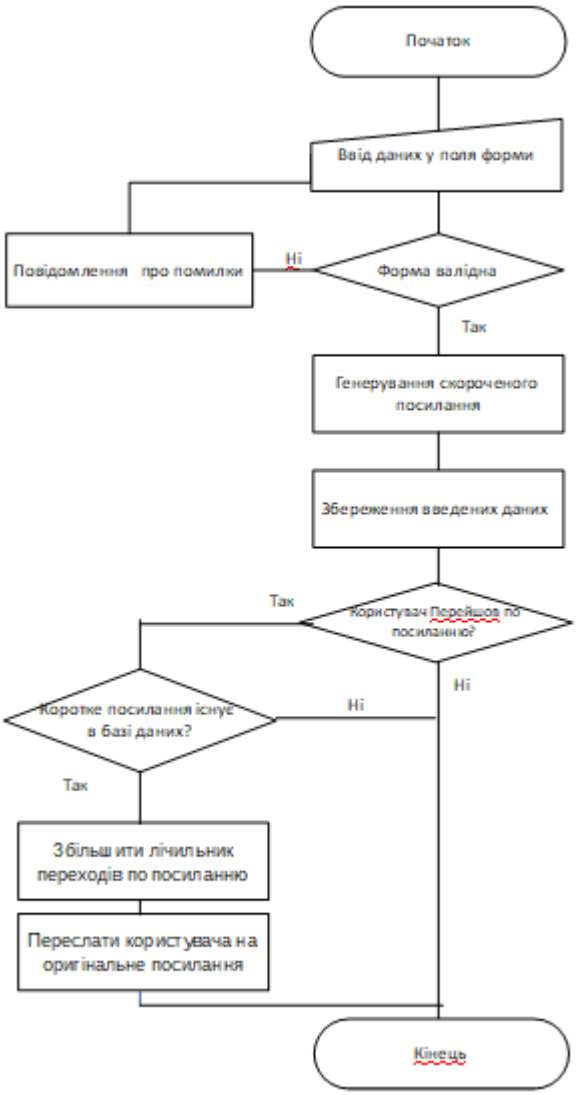
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бобкова О. К вопросу о соотношении понятий «доменное имя» и «название сайта» / О. Бобкова, С. К. Давидов., 2014
2. Что такое URL-адрес? [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/Understanding_URLs.
3. .Рой Ф. Architectural Styles and the Design of Network-based Software Architectures / Філдінг Рой., 2000.
4. How to Build a URL Shortener [Електронний ресурс] – Режим доступу до ресурсу: https://scalegrid.io/blog/how-to-build-a-url-shortener-with-node-js-and-mongodb.
5. Чем опасны сокращенные ссылки и как от этого защититься [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/hosting-cafe/blog/324092.
6. Документація Django [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/3.0/
7. Документація Django RestFramework [Електронний ресурс] – Режим доступу до ресурсу: https://www.django-rest-framework.org/
8. Документація Vue.JS [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/
9. Феокітцова О. Что такое SaaS и как это работает [Електронний ресурс] / ольга Феокітцова. – 2017. – Режим доступу до ресурсу: https://blog.ringostat.com/ru/chto-takoe-saas-i-kak-eto-rabotaet/.
10. Maynard J. Five Benefits of Software as a Service [Електронний ресурс] / Maynard. – 2007. – Режим доступу до ресурсу: www.trumba.com..
11. Hostiq. CMS - Що це Таке? [Електронний ресурс] / Hostiq. – 2018. – Режим доступу до ресурсу: https://hostiq.ua/wiki/ukr/cms-ukr/.
12. Почему Vue [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.vis-design.com/ru/blog/pochemu-vue-a-ne-react-ili-angular.html.

13. Одников В. Что стоит за простой загрузкой веб-странички в браузере [Електронний ресурс] / В. Одников. – 2018. – Режим доступу до ресурсу: https://bit.ly/3gRmLCW
14. Іващенко А. REST: простым языком [Електронний ресурс] / Андрій Іващенко. – 2019. – Режим доступу до ресурсу: https://bit.ly/2MuxTrp.
15. ТОП-10 фреймворков для веб-разработки в 2019 [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://proglib.io/p/web-frameworks-2019/.
16. 5 лучших сервисов для сокращения ссылок Источник: https://blog.uamaster.com/5-services-to-shorten-links/ - блог интернет-агентства UaMaster. [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://blog.uamaster.com/5-services-to-shorten-links/.
17. Smith L. Чем PostgreSQL лучше других SQL баз данных [Електронний ресурс] / Lisa Smith. – 2016. – Режим доступу до ресурсу: https://habr.com/ru/post/282764/.

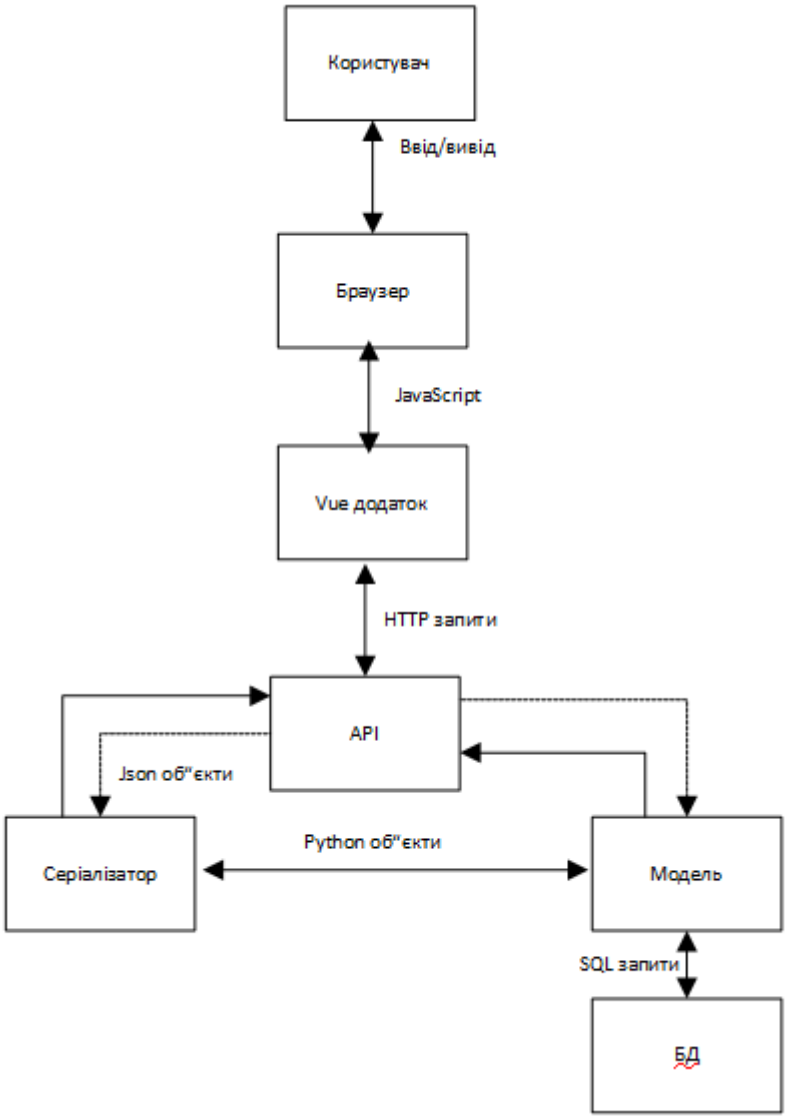
ДОДАТКИ

Схема Алгоритму



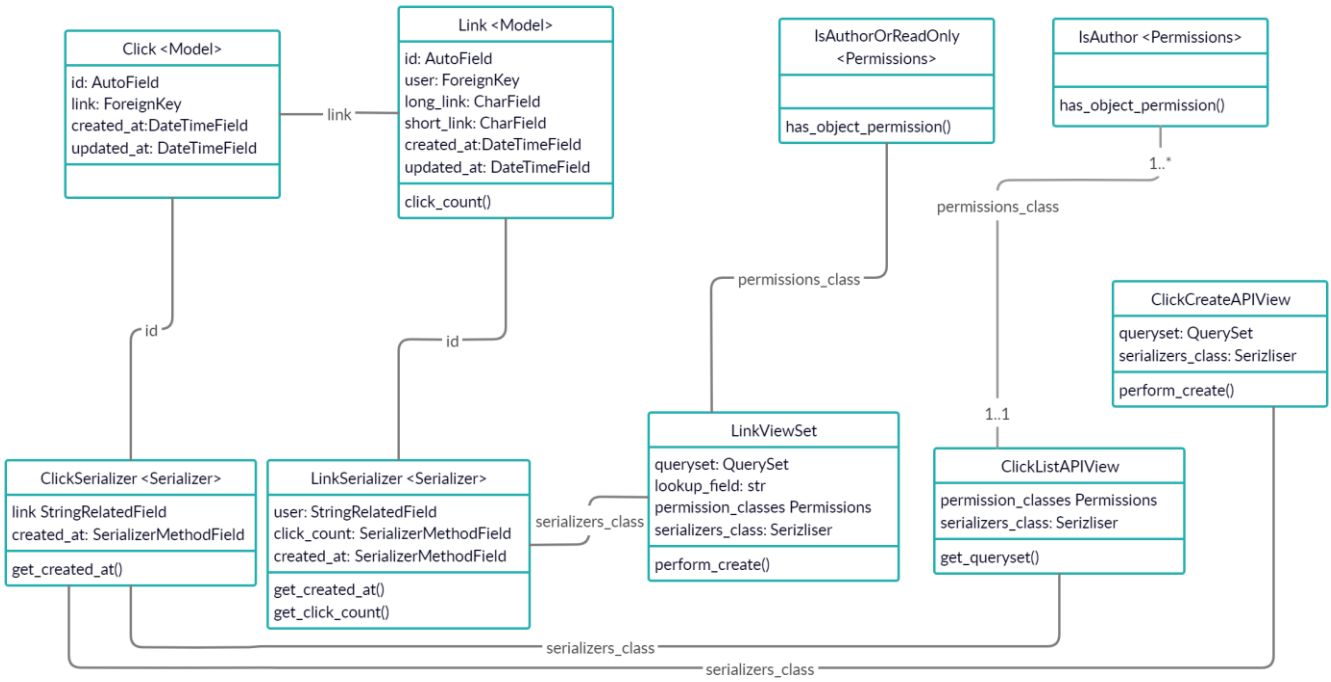
					ДП.6415.04.000 Д1		
Зм.		№ документа	Підп.	Дата			
					Сервіс для створення скорочених посилань	Літ.	Аркуш	
						Т	1	1
Н.конт		Симоненко В.П.				НТУУ «КПІ імені Ігора Сікорського», ФІОТ Група ІО - 64	
Затв.							

Функціональна Схема



					ДП.6415.05.000 Д2		
Зм.		№ документа	Підп.	Дата			
					Сервіс для створення скорочених посилань	Літ.	Аркуш / Аркушів
						Т	1 / 1
Н.конт		Симоненко В.П.				НТУУ «КПІ імені Ігора Сікорського», ФІОТ Група ІО - 64	
Затв.							

Діграма Класів



					ДП.6415.06.000 ДЗ		
Зм.		№ документа	Підп.	Дата			
					Сервіс для створення скорочених посилань	Літ.	Аркуш / Аркушів
						Т	1 / 1
Н.конт		Симоненко В.П.				НТУУ «КПІ імені Ігора Сікорського», ФІОТ Група ІО - 64	
Затв.							